**Batch: A1 Roll No.: 16010123012**

**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)**

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

**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.

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

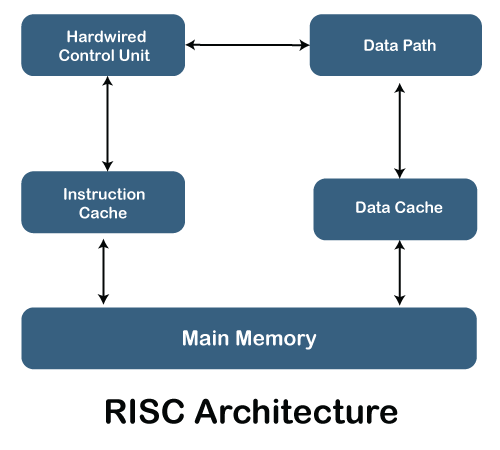
**Pre Lab/ Prior Concepts:**

**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:



1. Brief Explanation of each component
2. **Hardwired Control Unit**

* This is a control unit that generates control signals based on the hardwired logic. In RISC architecture, it facilitates quick and simple instructions by coordinating between the processor and the data path.

1. **Instruction Cache**

* A small, fast memory location that stores frequently accessed instructions to minimize the latency in fetching instructions from the main memory. It helps speed up the execution process by storing instructions close to the processor.

1. **Data Path**:

* The data path consists of the functional units (like ALU), registers and buses that carry data between them. It handles the processing of data and the execution of arithmetic and logical operations.

1. **Data Cache**:

* Similar to the instruction cache, the data cache stores frequently accessed data to reduce the delay of accessing data from main memory. It improves the performance of memory-bound operations.

1. **Main Memory**:

* This is the primary storage where instructions and data are stored. The processor fetches instructions from the instruction cache and data from the data cache. In case of a cache miss, data or instructions are fetched from the main memory.

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

### ****ADD (Addition Instruction) Syntax****: ADD Rd, Rs1, Rs2 Rd: Destination register where the result will be stored. Rs1: First source register that holds one operand. Rs2: Second source register that holds the other operand.

**Example**: ADD R1, R2, R3  
This instruction adds the values in registers R2 and R3, and stores the result in register R1.

**Explanation**:  
The **ADD** instruction performs integer addition between the contents of two registers and stores the result in a destination register.

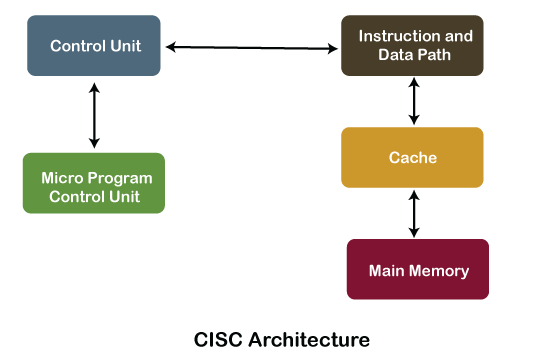
### ****LOAD (Load Word Instruction)** **Syntax****: LOAD Rd, offset(Rs) Rd: Destination register where the loaded word will be stored. offset: A constant value used to calculate the memory address. Rs: Base register that holds the memory address.

**Example**: LOAD R1, 4(R2)  
This instruction loads the word located at the memory address R2 + 4 into register R1.

**Explanation**:  
The **LOAD** instruction moves data from memory into a register. The address of the memory location is determined by adding the base address stored in Rs (here R2) and the offset (here 4).

**CISC Architecture**

1. Diagram of CISC Architecture:



1. Brief Explanation of each component

### ****Control Unit****

* The control unit is responsible for directing the flow of data and instructions within the processor. In CISC architecture, it decodes complex instructions and sends appropriate signals to other components (e.g., ALU, registers) to execute them.

### ****Microprogram Control Unit****

* This unit is part of the control system that uses a microprogram to generate the control signals necessary for executing complex instructions. Microprogramming allows a sequence of micro-operations to execute a single complex instruction.

### 3. ****Instruction and Data Path****

* This represents the pathways and mechanisms through which instructions and data flow through the processor. It includes components like the Arithmetic Logic Unit (ALU), registers, and buses that handle the execution of operations.

### 4. ****Cache****

* The cache stores frequently used instructions and data to improve the speed of execution. Since accessing main memory is slower, the cache ensures faster retrieval of data and instructions by holding copies closer to the CPU.

### 5. ****Main Memory****

* The main memory holds the instructions and data that the processor needs for execution. Instructions are fetched from the main memory if they are not available in the cache.

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

### ****MOV (Move Instruction)** **Syntax****: MOV destination, source destination: Can be a register or a memory location where the data will be moved. source: Can be a register, memory location, or immediate value from where the data will be moved.

**Example**: MOV AX, [1000H]  
This instruction moves the content at memory location 1000H into register AX.

**Explanation**:  
The **MOV** instruction is one of the most commonly used instructions in CISC processors. It moves data from one location to another. The MOV instruction in CISC can handle memory-to-memory, memory-to-register, and register-to-memory operations, making it highly versatile.

### ****MUL (Multiply Instruction) Syntax****: MUL operand operand: Can be a register or memory location containing the value to be multiplied.

**Example**: MUL BX  
This instruction multiplies the contents of register BX by the contents of the AX register (implicit operand) and stores the result in AX or DX:AX if the result is larger.

**Explanation**:  
The **MUL** instruction in CISC architecture allows direct multiplication of operands, and the result is stored automatically in dedicated registers (like AX and DX:AX).

**Post Lab Descriptive Questions**

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

|  |  |  |
| --- | --- | --- |
| **Basis for Comparison** | **RISC Architecture** | **CISC Architecture** |
| Emphasis | Software | Hardware |
| Instruction Size | Small | Large |
| Registers | More Registers Used | Less Registers Required |
| Pipelining | Easy | Difficult |
| Addressing Mode | Limited addressing mode required | More addressing mode required |

**Conclusion:** We have successfully completed this experiment and gained a detailed understanding of the architecture of RISC and CISC.

**Date: 07 / 09 / 2024 Signature of faculty in-charge**