Batch: D2	**Roll No.: 16010122323**
Experiment / assignment / tutorial No. 9	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Study of RISC and CISC Architecture

AIM: Understanding RISC and CISC Architechture

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)

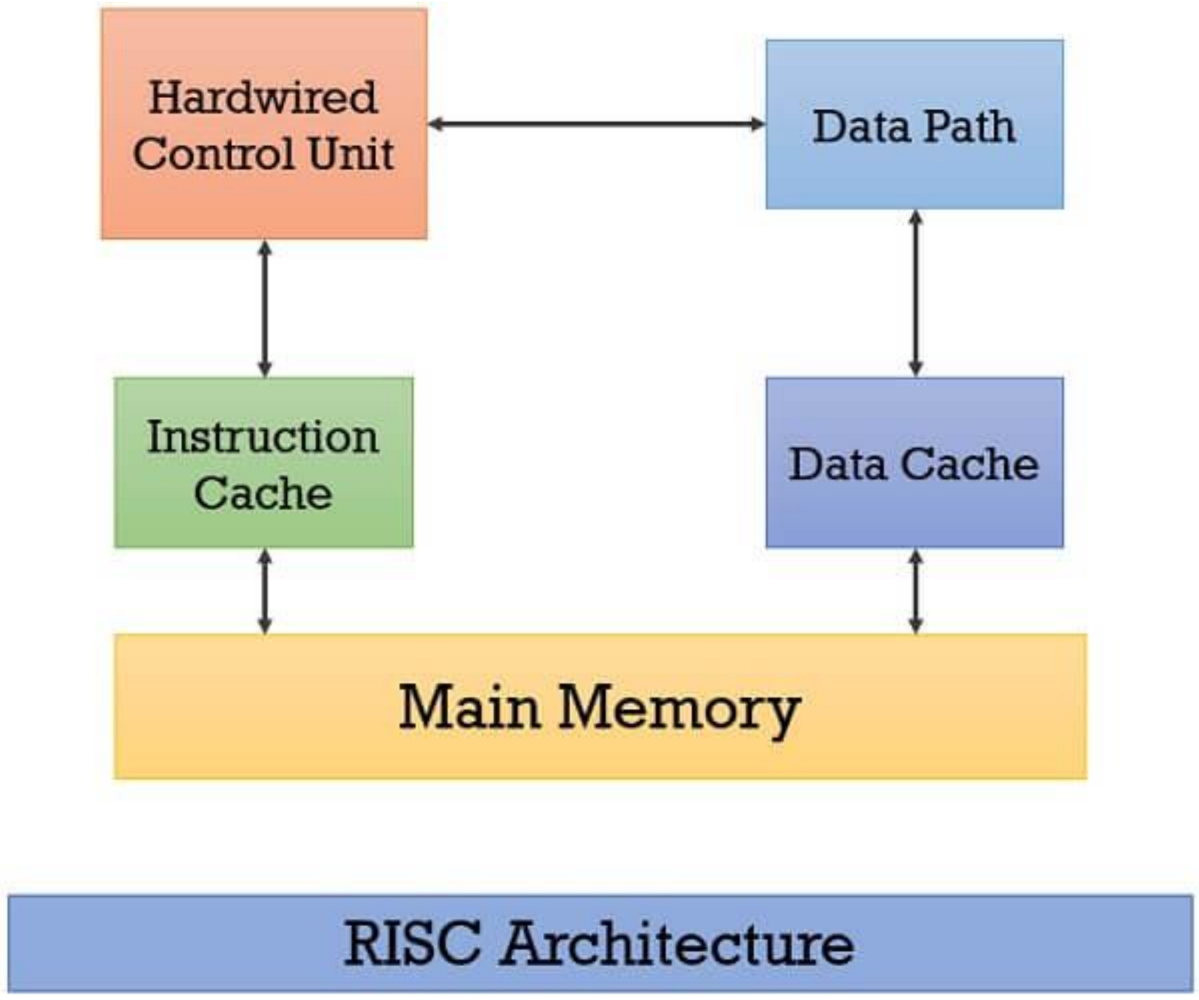
The main idea behind is to make hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating and storing operations just like a load command will load data, store command will store the data.

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:



2. Brief Explanation of each component

Hardwired control: To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence. Computer designers use a wide variety of techniques to solve this problem. The approaches used fall into one of two categories: hardwired control and microprogrammed control. The required control signals are determined by the following information:

- Contents of the control step counter
- Contents of the instruction register
- Contents of the condition code flags
- External input signals, such as MFC and interrupt requests

DataPath :

- A **datapath** is a collection of functional units such as arithmetic logic units or multipliers that perform data processing operations, registers, and buses.[1] Along with the control unit it composes the central processing unit (CPU).[1] A larger datapath can be made by joining more than one datapaths using multiplexers.
- A **data path** is the ALU, the set of registers, and the CPU's internal bus(es) that allow data to flow between them.
- A microarchitecture datapath organized around a single bus
- The simplest design for a CPU uses one common internal bus. Efficient addition requires a slightly more complicated three-internal-bus structure.[3] Many relatively simple CPUs have a 2-read, 1-write register file connected to the 2 inputs and 1 output of the ALU.

DataCache:

Cache is a small amount of memory which is a part of the CPU - closer to the CPU than RAM. It is used to temporarily hold instructions and data that the CPU is likely to reuse.

The CPU control unit automatically checks cache for instructions before requesting data from RAM. This saves fetching the instructions and data repeatedly from RAM – a relatively slow process which might otherwise keep the CPU waiting. Transfers to and from cache take less time than transfers to and from RAM.

The more cache there is, the more data can be stored closer to the CPU.

Cache is **graded** as Level 1 (L1), Level 2 (L2) and Level 3 (L3):

- **L1** is usually part of the CPU chip itself and is both the smallest and the fastest to access. Its size is often restricted to between 8 KB and 64 KB.
- **L2** and **L3** caches are bigger than **L1**. They are extra caches built between the CPU and the RAM. Sometimes L2 is built into the CPU with L1. L2 and L3 caches take slightly longer to access than L1. The more L2 and L3 memory available, the faster a computer can run.

Instruction cache :

Instructions supplied by the instruction cache are decoded by the instruction decoder, which sends the decoded operands and control information to the datapath. The data cache is used only for LOAD and STORE instructions (as with most RISC systems, F-RISC allows access to data memory only through these instructions.

Main Memory :

Memory Access is **accomplished through Load and Store instructions only**, thus the term "Load/Store Architecture" is often used when referring to RISC. The RISC pipeline is specified in a way in which it must accommodate both: operation and memory access with equal efficiency.

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

RISC instructions are simple and are of **fixed** size. Each RISC instruction engages a **single memory word**. RISC instructions operate on processor registers only. The instructions that have arithmetic and logic operation should have their operand either in the **processor register** or should be given **directly** in the instruction.
Like in both the instructions below we have the operands in registers

Add R2, R3

Add R2, R3, R4

The operand can be mentioned directly in the instruction as below:

Add R2, 100

But initially, at the **start of execution** of the program, all the **operands** are in **memory**. So, to access the memory operands, the RISC instruction set has **Load** and **Store** instruction.
The **Load** instruction loads the operand present in **memory** to the processor **register**. The load instruction is of the form:

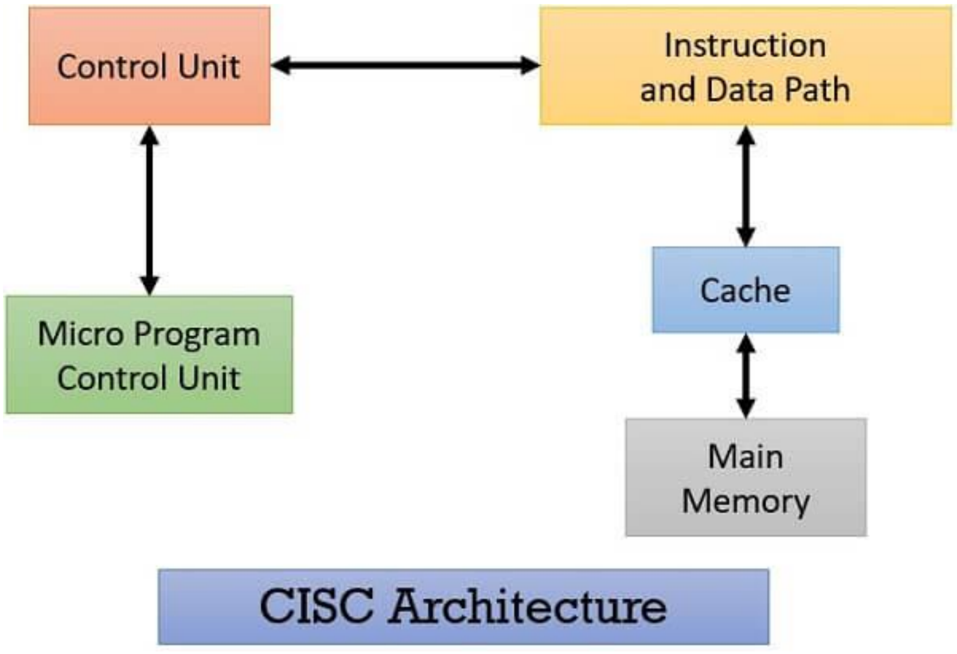
Load *destination, Source*

Example Load R2, A // *memory to register*
The load instruction above will load the operand present at memory location **A** to the processor register **R2**.
The Store instruction stores the operand back to the memory. Generally, the Store instruction is used to store the intermediate result or the final result in the memory. It is of the form:

Store *source, destination*n

CISC Architecture

1. Diagram of CISC Architecture:



2. Brief Explanation of each component

CONTROL UNIT :

The **control unit** (CU) is a component of a computer's central processing unit (CPU) that directs the operation of the processor. A CU typically uses a binary decoder to convert coded instructions into timing and control signals that direct the operation of the other units (memory, arithmetic logic unit and input and output devices, etc.).

Most computer resources are managed by the CU.

INSTRUCTION AND DATA PATH :

The datapath is the "brawn" of a processor, since it implements the fetch-decode-execute cycle. The general discipline for datapath design is to (1) determine the instruction classes and formats in the ISA, (2) design datapath components and interconnections for each instruction class or format, and (3) compose the datapath segments designed in Step 2) to yield a composite datapath.

Simple datapath components include *memory* (stores the current instruction), *PC* or program counter (stores the address of current instruction), and *ALU* (executes current instruction). The interconnection of these simple components to form a basic datapath

MICRO-PROGRAM CONTROL UNIT :

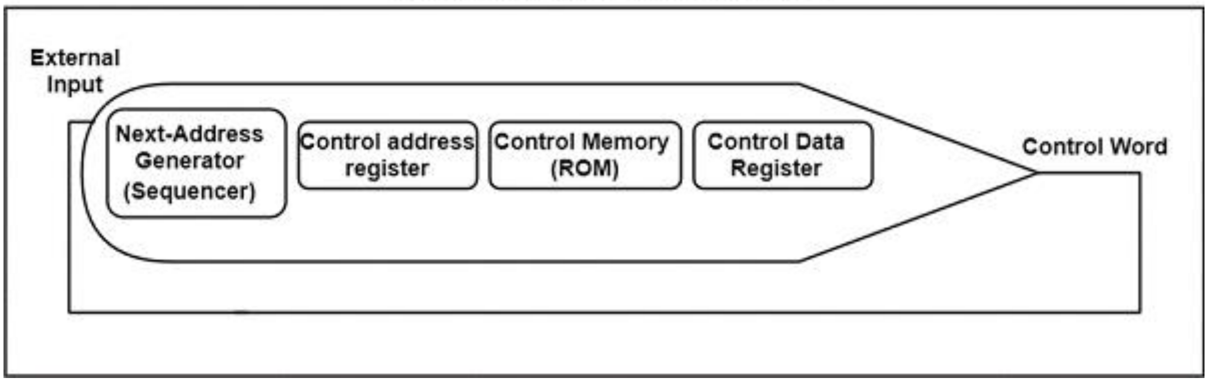
A control unit whose binary control values are saved as words in memory is called a microprogrammed control unit.

A controller results in the instructions to be implemented by constructing a definite collection of signals at each system clock beat. Each of these output signals generates one micro-operation including register transfer. Thus, the sets of control signals are generated definite micro-operations that can be saved in the memory.

Each bit that forms the microinstruction is linked to one control signal. When the bit is set, the control signal is active. When it is cleared the control signal turns inactive. These microinstructions in a sequence can be saved in the internal ’control’ memory. The control unit of a microprogram-controlled computer is a computer inside a computer.

The following image shows the block diagram of a Microprogrammed Control organization.

Micro Programmed Control Organization



There are the following steps followed by the microprogrammed control are −

- It can execute any instruction. The CPU should divide it down into a set of sequential operations. This set of operations are called microinstruction. The sequential micro-operations need the control signals to execute.
- Control signals saved in the ROM are created to execute the instructions on the data direction. These control signals can control the micro-operations concerned with a microinstruction that is to be performed at any time step.
- The address of the microinstruction is executed next is generated.

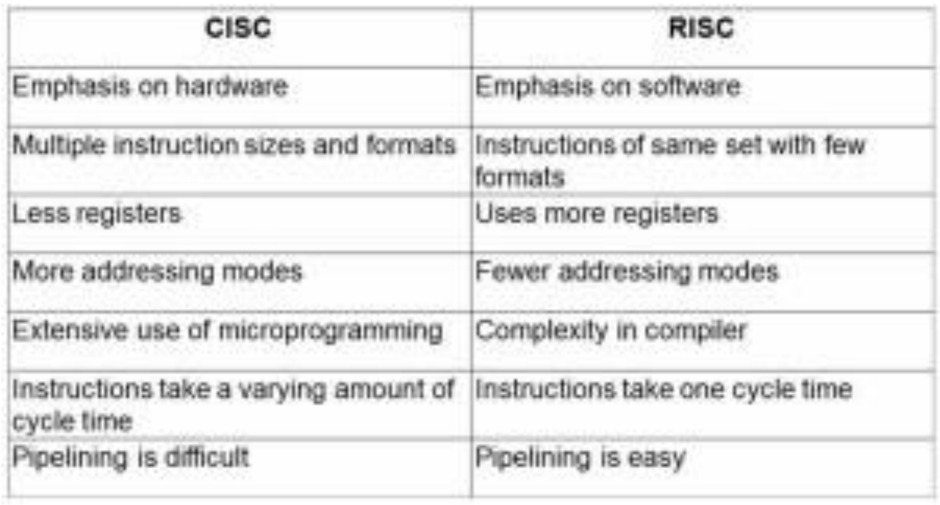
3. CISC Processor Instruction Set Examples with explanation (Any 2)

The decoding of instructions is simple. Examples of CISC processors are the System/360, VAX, AMD, and Intel x86 CPUs. Common RISC microprocessors are ARC, Alpha, ARC, ARM, AVR, PA-RISC, and SPARC. Important applications are Security systems, Home automation

Post Lab Descriptive Questions

Write a tabular comparative analysis of RISC v/s CISC

- The wasting cycles can be prevented by the programmer by removing the unnecessary code in the RISC, but, while using the CISC code leads to wasting cycles because of the inefficiency of the CISC.
- In RISC, each instruction is intended to perform a small task such that, to perform a complex task, multiple small instruction are used together, whereas only few instructions are required to do the same task using CISC – as it is capable of performing complex task as the instructions are similar to a high-language code.
- CISC is typically used for computers while RISC is used for smart phones, tablets and other electronic devices.



CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy