RISC and CISC Architecture

Reduced Instruction Set Computer or RISC Architecture

The fundamental goal of RISC is to make hardware simpler by employing an instruction set that consists of only a few basic steps used for evaluating, loading, and storing operations. A load command loads data but a store command stores data.

Characteristics of RISC:

1. It has simpler instructions and thus simple instruction decoding.

2. More general-purpose registers.

3. The instruction takes one clock cycle in order to get executed.

4. The instruction comes under the size of a single word.

5. Pipeline can be easily achieved.

6. Few data types.

7. Simpler addressing modes.

Complex Instruction Set Computer or CISC Architecture

The fundamental goal of CISC is that a single instruction will handle all evaluating, loading, and storing operations, similar to how a multiplication command will handle evaluating, loading, and storing data, which is why it’s complicated.

Characteristics of CISC:

1. Instructions are complex, and thus it has complex instruction decoding.

2. The instructions may take more than one clock cycle in order to get executed.

3. The instruction is larger than one-word size.

4. Lesser general-purpose registers since the operations get performed only in the memory.

5. More data types.

6. Complex addressing modes.

Both CISC and RISC approaches primarily try to increase the performance of a CPU. Here is how both of these work:

1. CISC: This kind of approach tries to minimize the total number of instructions per program, and it does so at the cost of increasing the total number of cycles per instruction.

2. RISC: It reduces the cycles per instruction and does so at the cost of the total number of instructions per program.

Example

Suppose we need to add two different 8-bit numbers:

1. CISC approach: There would be a single instruction or command for this, such as ADD, that would perform the task.

2. RISC approach: In this case, the programmer would write the very first load command in order to load data in the registers. Then it would use a suitable operator and store the obtained result in the location that is desired.

The add operation here is divided into parts, namely, operate, load, and store. Due to this, RISC programs are much longer, and they require more memory to get stored, even though they require fewer transistors because the commands are less complex.