**CISC**

CISC stands for Complex Instruction Set Computer chips, which are memory-efficient and simple to program. The CISC philosophy was frequently used in large computers like the PDP-11 and the DEC system 10 and 20 machines because the initial machines were written in assembly language and memory was slow and expensive.

To make compiler development easier, CISC was created. It transfers the majority of the responsibility for creating machine instructions to the processor. For instance, a CISC processor would already be able to calculate a square root without the need for a compiler to construct complicated machine instructions.

The CISC technique aims to cut down on the number of cycles per instruction by lowering the number of instructions per program. The CISC architecture is used to build computers that have lower memory costs. Huge programs require more storage, which raises the cost of memory as well as the price of large memory. In order to address these problems, it is possible to reduce the number of instructions needed for each system by consolidating the number of operations into a single instruction, increasing the complexity of the instructions in the process.

CISC instruction sets share the following traits:

• A two-operand format where the source and destination of the instructions are different. Commands can be registered to memory, registered to register, and registered to register.

• Instructions of variable lengths, whose lengths frequently change depending on the addressing method.

• Instructions that must be executed across a number of clock cycles.

Due to the requirement for a single instruction to support several addressing modes, complex instruction-decoding circuitry is required.

• A few registers that are general-purpose. This is due to the restricted amount of chip area not devoted to instruction decoding, execution, and microcode storage as well as the existence of instructions that can be executed directly on memory.

• Several registries for various purposes. For the stack pointer, interrupt management, and other purposes, specific registers are often set in CISC architectures.

• A register called "Condition code" that is set by most instructions. This register keeps track of specific error circumstances and indicates whether the outcome of the most recent operation is less than, equal to, or higher than zero.

When they were first developed, CISC machines made use of current technologies to boost computer performance. Microprogramming is substantially less expensive than hardwiring a control unit and is just as simple to implement as assembly language.

• Because it was simple to micro-code new instructions, designers were able to create CISC machines that were upwardly compatible, meaning that a new computer could execute the same programs as older ones because it would have a superset of the older computers' instructions.

• Fewer instructions might be used to complete a task as each instruction gained in capability. This utilized the relatively slow main memory more effectively.

• The compiler does not need to be as complex because microprogram instruction sets can be created to correspond to the features of high-level languages.

Principal features of a CISC design »1) A large number of instructions, often between 100 and 250

2) A few infrequently used instructions that carry out particular duties

3) A wide range of addressing options

4) Formats for instruction with variable length

5) Memory-based operand manipulation instructions

**Addressing Modes in CISC**

2. Variable-length instructions are a result of CISC processor designers' choice to offer a range of addressing options. For instance, if an operand is in memory rather than a register, the length of the instruction rises.

This is due to the fact that memory addresses must be specified as part of instruction encoding, which requires many more bits.

b. This makes scheduling and decoding of instructions more difficult. Because there are so many different instruction types available, different amounts of clocks are needed to perform each instruction.

c. This once more causes issues with lesson planning and scheduling.

**Advantages of CISC Processors**

1. High-level programs or statement languages are easily converted into assembly or machine language in CISC processors by the compiler.

2. The code is rather small, which reduces the amount of RAM needed.

3. There is a very small RAM need for each CISC to store an instruction.

4. A single instruction needs the completion of numerous low-level operations.

5. To control power consumption, CISC develops a procedure that modifies voltage and clock speed.

6. It needs fewer instructions than the RISC to carry out the identical task.

**Disadvantages of CISC Processors**

1. CISC chips are slower than RISC chips to execute per instruction cycle on each program.

2. The performance of the machine decreases due to the slowness of the clock speed.

3. Executing the pipeline in the CISC processor makes it complicated to use.

4. The CISC chips require more transistors as compared to RISC design.

5. In CISC it uses only 20% of existing instructions in a programming event.

**Examples of CISC:**

**System/360**

The Variant 30, the slowest System/360 model introduced in 1964, could execute up to 34,500 instructions per second and had memory ranging from 8 to 64 KB. Later came high-performance models. Up to 16.6 million instructions may be processed every second by the 1967 IBM System/360 Model 91 computer. A big installation could have as little as 256 KB of main storage, but 512 KB, 768 KB, or 1024 KB was more typical. The larger 360 models could have up to 8 MB of main memory, however that much main memory was exceptional. For some models, slower (8 microsecond) Large Capacity Storage (LCS) of up to 8 megabytes was also an option.

**Motorola 6800**.

The Motorola 6800 is an 8-bit microprocessor that was created and released in 1974. The M6800 Microcomputer System, which also comprised RAM, ROM, parallel and serial interface ICs, and other support chips, had the MC6800 CPU. The M6800 family of ICs' single five-volt power supply need, at a time when the majority of other microprocessors required three voltages, was an important design element. The 6800 includes an 8-bit bi-directional data bus and a 16-bit address bus that can directly access 64 KB of memory. It includes 197 opcodes altogether and 72 instructions with seven addressing modes. Clock frequency for the original MC6800 was up to 1 MHz. Later models have a 2 MHz maximum clock frequency.

Motorola offered an entire assembly language development system in addition to the ICs. The customer might utilize the software on a local minicomputer system or a remote timeshare computer. The M6800 ICs were used to construct the Motorola Exorciser, a desktop computer that could be used for developing and debugging new designs. Datasheets for every IC were supplied in a comprehensive documentation package, together with two guides on programming in assembly language and a 700-page application guide that explained how to construct a point-of-sale terminal—a computerized cash register—based on the 6800.

**RISC**

Reduced instruction set computer, or RISC, is a type of microprocessor created to carry out fewer different kinds of computer instructions so that it can run at a faster rate (perform more millions of instructions per second, or MIPS). A larger list or set of computer instructions tends to make the microprocessor more complex and slower to operate since each instruction type that a computer must do requires additional transistors and circuitry.

The RISC concept was developed in 1974 by John Cocke of IBM Research in Yorktown, New York, who demonstrated that around 20% of a computer's instructions performed 80% of the work. The IBM PC/XT was the first computer to use this breakthrough in 1980. The RISC idea has encouraged a more careful approach to microprocessor design. Design factors include how successfully an instruction can be mapped to the microprocessor's clock speed (ideally, an instruction should be completed in one clock cycle), how basic the needed architecture is, and how much work the microchip itself can complete without the aid of software.

There are some design elements that set it apart from the competition:

• **Execution time for one cycle**. CPI (clock per instruction) for RISC processors is one cycle. This is caused by the CPU's optimization of every instruction and a process known as pipelining.

• **Pipelining**. a method for processing instructions more quickly that enables the simultaneous execution of different parts or stages of an instruction.

• There are a lot of registers. A greater number of registers are typically used in RISC design in order to reduce the quantity of memory interactions.

**Typical Characteristics of RISC Architecture**

The key findings that encouraged designers to think about CISC design possibilities were:

a. **Simple guidelines** because they bridge the semantic gap, sophisticated instructions were heavily anticipated by CISC architectural designers. It turns out that compilers largely disregard these directives in practice. This has been demonstrated by a number of empirical research. The usage of various meanings by high-level languages is one explanation for this. For instance, the C for loop's semantics differ slightly from those of other languages. As a result, compilers frequently use simpler instructions to create the code.

b. **A Few Data Types**: the CISC ISA typically supports a wide range of data structures, from straightforward data types like characters and integers to more intricate ones like records and structures. According to empirical data, sophisticated data structures are utilized only sometimes.

Therefore, it is advantageous to build a system that supports a few simple data types effectively and allows for the synthesis of the complicated data kinds that are missing.

c**. Simple Addressing Modes**: many addressing modes are offered by CISC designs. The two main drivers are to:

* support complicated data structures and
* offer operand access flexibility.

1. Issues Raised. This offers flexibility, but it also creates issues. First, depending on where the operands are located, it results in varied instruction execution times.
2. It results in variable-length instructions, secondly. The IA-32 instruction length, for instance, can vary from 1 to 12 bytes. Variable instruction lengths result in ineffective scheduling and decoding of instructions.

d. Exactly the same general-purpose registers. the ability to use any register in any situation, simplifying compiler design (even if there are typically separate floating-point registers).

e. Harvard-based architecture. Because the CPU has a separate instruction and data cache, changing the memory where code is stored might not have any impact on the instructions that are executed by the processor—at least not until a special synchronization instruction is issued. RISC designs are also more likely to use a Harvard memory model, where the instruction stream and the data stream are conceptually separated. On the plus side, this enables simultaneous access to both caches, which frequently enhances performance.

### Advantages of RISC Processor

• If making a new microprocessor simpler is one of its goals, this can speed up the development and testing process.

• Operating system and application programmers will find it simpler to write code with a reduced instruction set if they employ the microprocessor's instructions.

• Higher-level language compilers now write more efficient code than in the past since they have always tended to use the smaller set of instructions to be found on a RISC machine, which is made possible by the simplicity of RISC.

• The cost of RAM has dropped significantly. 1MB of DRAM cost roughly $5,000 in 1977.

• In 1994, the identical amount of RAM only cost $6. (When adjusted for inflation). The RISC usage of RAM and concentration on software has become perfect due to the advancement of compiler technology.

**Disadvantages of RISC Processor**

1. The performance of the RISC processor might vary depending on the code that is run because a cycle's worth of instructions may depend on one another.

2. Complex instructions are frequently used by programmers and compilers.

3. RISC processors need highly quick memory to store a variety of instructions that need a lot of cache memory to react to the command quickly.

**RISC Processors (Examples)**

**Digital Equipment Corporation (DEC)** - Alpha, formerly known as Alpha AXP, is a 64-bit RISC instruction set architecture (ISA) that was created by DEC to take the role of the 32-bit VAX complex instruction set computer (CISC) ISA and its implementations.

a. DEC was the company that first designed and produced the microprocessors that used Alpha.

b. These microprocessors were primarily utilized in a number of DEC workstations and servers, which eventually served as the foundation for nearly all of their mid-to-high-end product lines.

c. A number of independent suppliers also created Alpha systems, including motherboards in the PC form factor.

**SuperH**

Hitachi created SuperH (SH), a RISC instruction set architecture (ISA) for 32-bit computers. Microprocessors and microcontrollers used in embedded systems implement it. It falls into the following main categories:

a. SH-1, which is used in microcontrollers for highly embedded applications (CD- ROM drives, major appliances, etc.)

b. SH-2. Used in microcontrollers with higher performance requirements, as well as in networking applications or engine control units in automobiles, as well as in gaming consoles like the Sega Saturn. The SH-2 has also been used in numerous applications involving motor control.

c. SH-DSP, originally created for the market for mobile phones, afterwards utilized in numerous consumer applications needing DSP performance for JPEG compression, etc.

d. SH-3, which has a long history in the market for automotive navigation systems and is used for mobile and handheld devices like the Jornada.

e. SH-3 DSP. used mostly in networking applications and multimedia interfaces, as well as in printers and fax machines.

f. SH-4. Used in applications where high performance is required, such as set-top boxes, video game consoles, and automobile multimedia terminals.

g. SH-5 is used in high-end multimedia applications.

**RISC vs. CISC**

Comparison of the two using an example:

Memory Multiplication of Two Numbers Locations in the main memory are numbered from (row) 1: (column) 1 to (row) 6: (column) 4. All computations must be performed by the execution unit. But only data that has been loaded into one of the six registers can be used by the execution unit (A, B, C, D, E, or F). Let's imagine we want to determine the product of two numbers that are stored at locations 2:3 and 5:2, then store the result back at position 2:3.

**The CISC Approach**: CISC architecture's main objective is to use the least number of assembly lines necessary to execute a task. This is accomplished by designing processing hardware that can comprehend and carry out a number of functions. A CISC processor would be equipped with a special instruction for this task (say "MUL").

a. This instruction multiplies the operands in the execution unit before storing the result in the appropriate register after loading the two values into separate registers.

b. As a result, just one instruction is required to multiply two numbers:

MUL 2:3, 5:2

c. MUL falls under the category of "complex instruction."

d. It doesn't require the programmer to explicitly call any loading or saving functions; instead, it operates directly on the memory banks of the computer.

e. It resembles a command in a higher-level language in a lot of ways. This command is equivalent to the C phrase "a = an x b," for example, if we let "a" represent the value of 2:3 and "b" represent the value of 5:2.

**The RISC Approach**: only straightforward commands that can be completed in a single clock cycle are used by RISC processors. As a result, the command "MUL" mentioned above could be broken into three different commands:

A "LOAD" command loads data into a register; a "PROD" command calculates the product of two operands inside a register; and a "STORE" command transfers data from a register to the memory banks.

d. A programmer would need to write four lines of assembly code in order to carry out the precise succession of steps outlined in the CISC approach:

2:3 LOAD A 5:2 LOAD B

A, B, PROD, STORE 2:3, A