Instruction set Based classification of processors

(RISC, CISC, and their Comparison)

Instruction Set Architecture (ISA) is the set of processor design techniques used to implement the instruction work flow on hardware.

"ISA tells that how your processor going to process your program instructions."

What is RISC? A Reduced Instruction set Computer is a computer which only use simple instructions that can be divide into multiple instructions which perform low level operation with single clock-cycle.

What is CISC? A complex instruction set Computer is a computer where single instructions can execute several low-level operations (such as load from memory, an arithmetic operation, and a memory store) or are capable of multistep operations or addressing modes within single instructions

RISC & CISC ARCHITECTURE WITH EXAMPLE

Let we take an example of Multiplying two numbers:-

A = A * B <—— this is C Statement

CISC Approach:-

→ CISC processor would come prepared with a Specific instruction (We'll call it 'MULT'). When executed, this instruction

1. Loads the two values into Separate Registers.
2. Multiplies the operands in the execution unit.
3. And finally third, stores the product in the appropriate Register.

Thus the entire task of multiplying two no. Can be Completed with one instruction.

MULT A,B <<=== this is assembly Statement

** Mult is what is known as a Complex "Instruction." It operates directly on the Computer's memory banks and does not require any programmer to explicitly call any loading or storing functions.

Advantage:-

1⇒ Compiler has to do very little work to translate a high level language Statement into Assembly Language.

2⇒ Length of the code is relatively short.

3⇒ Very little RAM is Required to store instructions.

4⇒ The emphasis is put on building complex instructions directly into hardware.

The RISC Approach:- RISC only use simple instructions that can be executed with in one clock cycle. Thus the 'MULT' command described in CISC could be divided into three separate commands.

1. "LOAD" = Moves data from memory bank to a register.
2. "PROD" = which finds the product of two operands located within Registers.
3. Store :- which moves data from a Register to the memory banks.

In order to perform the exact series of steps described in the CISC approach, a programmer would need to code four lines of Assembly.

```
LOAD R1, A
LOAD R2, B
PROD A, B
STORE R3, A
```


At first, this may seem like a much less efficient way of completing the operation. Because there are more lines of code more RAM is needed to store the assembly level instructions. The Compiler must also perform more work to convert a high level language statement into code of this form.

Advantage:-

1) Each instruction requires only one clock cycle to execute, the entire program will execute in approximately the same amount of time as the multi-cycle- MULT command.

2) These RISC "Reduced Instructions" require less transistors of hardware space than the complex instructions, leaving more Room for General purpose Registers. Because all of the instructions execute in a uniform amount of time (i.e one clock)

3) Pipeline is possible

*

→ LOAD|STORE Mechanism:- Separating the "LOAD" and "Store" instructions actually reduces the amount of work that the Computer must perform.

→ After CISC style MULT command is executed, the processor automatically erases the registers.

→ If one of the operand needs to be used for another Computation, the processor must load again the data from memory to register.

→ In RISC, the Operand will remain in the Register until another value is loaded in its place.

Comparison of RISC & CISC

CISC	RISC
→ Emphasis on Hardware	Emphasis on Software
→ Includes multi-clock	Single clock
→ Complex instructions	Reduced instruction only
→ Memory-to Memory : "LOAD" and "Store" Incorporated in Instrn.	Register to Register : "LOAD" and "Store" are independent instructions
→ High Cycles per Second, Small Code Sizes	Low Cycles per second, large Code Sizes.
→ Transistors used for Storing Complex instructions	Spends more transistors on memory Registers.
Ex. PDP-II, VAx, Motorola 68K Intel's x86	DEC Alpha, AMD 29K Motorola 88000, SPARC