Instruction Set Architecture (ISA)

Instruction set architecture (ISA) is the set of processor design techniques used to implement the instruction work flow on hardware. In more practical words, ISA tells you that how your processor going to process your program instructions.

Complex Instruction Set Computer (CISC)

A complex instruction set computer (CISC) is a computer where single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions, as its name suggest "COMPLEX INSTRUCTION SET".

Examples of CISC instruction set architectures are PDP-11, VAX, Motorola 68k, and your desktop PCs on intel's x86 architecture based too.

Reduced Instruction Set Computer (RISC)

A reduced instruction set computer (RISC) is a computer which only use simple instructions that can be divide into multiple instructions which perform low-level operation within single clock cycle, as its name suggest "REDUCED INSTRUCTION SET"

Examples of RISC families include DEC Alpha, AMD 29k, ARC, Atmel AVR, Blackfin, Intel i860 and i960, MIPS, Motorola 88000, PA-RISC, Power (including PowerPC), SuperH, SPARC and ARM too.

Understand RISC & CISC architecture with example

Let we take an example of multiplying two numbers

```
A = A * B; <<<======this is C statement
```

The CISC Approach :- The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible. This is achieved by building processor hardware that is capable of understanding & executing a series of operations, this is where our CISC architecture introduced .

For this particular task, a CISC processor would come prepared with a specific instruction (we'll call it "MULT"). When executed, this instruction

- Loads the two values into separate registers
- Multiplies the operands in the execution unit
- And finally third, stores the product in the appropriate register.

Thus, the entire task of multiplying two numbers 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 the programmer to explicitly call any loading or storing functions.

Advantage:-

- Compiler has to do very little work to translate a high-level language statement into assembly
- Length of the code is relatively short
- Very little RAM is required to store instructions
- The emphasis is put on building complex instructions directly into the hardware.

The RISC Approach :- RISC processors only use simple instructions that can be executed within one clock cycle. Thus, the “MULT” command described above could be divided into three separate commands:

- “LOAD” which moves data from the memory bank to a register
- “PROD” which finds the product of two operands located within the registers
- “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      <<<======this is assembly statement

LOAD R2,B       <<<======this is assembly statement

PROD A, B       <<<======this is assembly statement

STORE R3, A     <<<======this is assembly statement
```

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.

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.

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)

Pipelining is possible.

LOAD/STORE mechanism:- Separating the “LOAD” and “STORE” instructions actually reduces the amount of work that the computer must perform. After a CISC-style “MULT” command is executed, the processor automatically erases the registers. If one of the operands needs to be used for another computation, the processor must re-load the data from the memory bank into a register. In RISC, the operand will remain in the register until another value is loaded in its place.

Comparison of RISC & CISC

RISC	**CISC**
1. RISC stands for Reduced Instruction Set Computer.	1. CISC stands for Complex Instruction Set Computer.
2. RISC processors have simple instructions taking about one clock cycle. The average clock cycle per instruction (CPI) is 1.5	2. CSIC processor has complex instructions that take up multiple clocks for execution. The average clock cycle per instruction (CPI) is in the range of 2 and 15.
3. Performance is optimized with more focus on software	3. Performance is optimized with more focus on hardware.
4. It has no memory unit and uses a separate hardware to implement instructions..	4. It has a memory unit to implement complex instructions.
5. It has a hard-wired unit of programming.	5. It has a microprogramming unit.
6. The instruction set is reduced i.e. it has only a few instructions in the instruction set. Many of these instructions are very primitive.	6. The instruction set has a variety of different instructions that can be used for complex operations.
7. The instruction set has a variety of different instructions that can be used for complex operations.	7. CISC has many different addressing modes and can thus be used to represent higher-level programming language statements more efficiently.
8. Complex addressing modes are synthesized using the software.	8. CISC already supports complex addressing modes
9. Multiple register sets are present	9. Only has a single register set
10. RISC processors are highly pipelined	10. They are normally not pipelined or less pipelined
11. The complexity of RISC lies with the compiler that executes the program	11. The complexity lies in the microprogram
12. Execution time is very less	12. Execution time is very high
13. Code expansion can be a problem	13. Code expansion is not a problem
14. Decoding of instructions is simple.	14. Decoding of instructions is complex
15. It does not require external memory for	15. It requires external memory for calculations

calculations	
16. The most common RISC microprocessors are Alpha, ARC, ARM, AVR, MIPS, PA-RISC, PIC, Power Architecture, and SPARC.	16. Examples of CISC processors are the System/360, VAX, PDP-11, Motorola 68000 family, AMD and Intel x86 CPUs.
17. RISC architecture is used in high-end applications such as video processing, telecommunications and image processing.	17. CISC architecture is used in low-end applications such as security systems, home automation, etc.

Which one is better ?

We can not differentiate RISC and CISC technology because both are suitable at its specific application. What counts is how fast a chip can execute the instructions it is given and how well it runs existing software. Today, both RISC and CISC manufacturers are doing everything to get an edge on the competition.

What’s new ?

You might thinking that RISC is nowdays used in microcontroller application widely so its better for that particular application and CISC at desktop application. But reality is boh are at threat position cause of a new technology called EPIC.

EPIC (Explicitly Parallel Instruction Computing) :-EPIC is a invented by Intel and is in a way, a combination of both CISC and RISC. This will in theory allow the processing of Windows-based as well as UNIX-based applications by the same CPU.