- **CISC**

Il CISC è la filosofia di costruzione dei microprocessori che mira ad avere un ISA complessa ossia con più istruzioni. Ciò permette una maggiore facilità di programmazione e di debugging (in assembler).

di seguito viene riportata l'equazione fondamentale per i processori:

$$\frac{tempo}{programma} = \frac{istruzioni}{programma} \times \frac{cicli}{istruzione} \times \frac{tempo}{ciclo}$$

vantaggi: questo comporta un minor costo di programmazione ed inoltre permette di avere un codice più denso e che occupa meno una memoria.

svantaggi: i processori CISC tendono a diminuire il rapporto istruzioni/programma cioè il numero di istruzioni che accompagnano il programma.

Tuttavia se il primo termine scende può avvenire che gli altri due aumentino a dismisura. questo poiché:

1.Le istruzioni contengono un sacco di store e di load che sono accessi in memoria che rallentano sistema.

2.Le istruzioni sono complesse e vanno decodificate prima di essere eseguite. Per questo viene utilizzata nei CISC una ROM di decodifica.

Il problema dei CISC resta sempre lo stesso; la CPU spreca un sacco di cicli per la decodifica.

RISC

Innanzitutto è necessario definire ciò che viene chiamato cammino critico (critical path):

" per cammino critico si intende il percorso più lungo che un segnale deve attraversare nel tempo di un ciclo di clock ".

un sistema per ovviare al critical path e la pipeline nata appunto assieme all'approccio RISC per la progettazione del processo.

L'approccio RISC:

1. Oltre il 90% del tempo il processore utilizza sempre un ristretto sottoinsieme di istruzioni. Il RISC possiede un numero ridotto di istruzioni semplici. In tal modo torna in auge il ruolo del compilatore il quale deve spezzettare le istruzioni più complesse istruzioni più semplici permettendo una esecuzione diretta senza interpretazione.

2. l'approccio RISC cerca di fare in modo che ogni istruzione semplice venga eseguita in un solo ciclo di clock (situazione ideale).

3. inoltre l'esecuzione dei programmi è rallentata dai ripetuti accessi di memoria centrale.Nel RISC l'accesso in memoria avviene esclusivamente tramite due comandi: load e store.

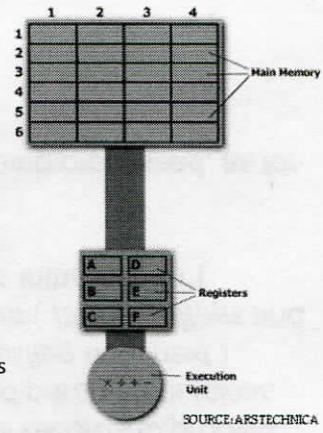
Note sulla formula per i RISC:

$\frac{tempo}{ciclo}$ tende a diminuire nel RISC poiché è più breve il percorso critico

$\frac{istruzioni}{programma}$ ovviamente questo fattore peggiora per la stessa filosofia RISC

$\frac{cicli}{isruzione}$ questo valore idealmente è molto vicino a 1

The simplest way to examine the advantages and disadvantages of RISC architecture is by contrasting it with it's predecessor: CISC (Complex Instruction Set Computers) architecture.



Multiplying Two Numbers in Memory

On the right is a diagram representing the storage scheme for a generic computer. The main memory is divided into locations numbered from (row) 1: (column) 1 to (row) 6: (column) 4. The execution unit is responsible for carrying out all computations. However, the execution unit can only operate on data that has been loaded into one of the six registers (A, B, C, D, E, or F). Let's say we want to find the product of two numbers - one stored in location 2:3 and another stored in location 5:2 - and then store the product back in the location 2:3.

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 and executing a series of operations. 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 then stores the product in the appropriate register. Thus, the entire task of multiplying two numbers can be completed with one instruction:

MULT 2:3, 5:2

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. It closely resembles a command in a higher level language. For instance, if we let "a" represent the value of 2:3 and "b" represent the value of 5:2, then this command is identical to the C statement "a = a * b."

One of the primary advantages of this system is that the compiler has to do very little work to translate a high-level language statement into assembly. Because the 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, and "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 A, 2:3
LOAD B, 5:2
PROD A, B
STORE 2:3, 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.

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock complex instructions	Single-clock, reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
Small code sizes, high cycles per second	Low cycles per second, large code sizes
Transistors used for storing complex instructions	Spends more transistors on memory registers

However, the RISC strategy also brings some very important advantages. Because 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.

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.

The Performance Equation

The following equation is commonly used for expressing a computer's performance ability: $\frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$

The CISC approach attempts to minimize the number of instructions per program, sacrificing the number of cycles per instruction. RISC does the opposite, reducing the cycles per instruction at the cost of the number of instructions per program.

RISC Roadblocks

Despite the advantages of RISC based processing, RISC chips took over a decade to gain a foothold in the commercial world. This was largely due to a lack of software support.

Although Apple's Power Macintosh line featured RISC-based chips and Windows NT was RISC compatible, Windows 3.1 and Windows 95 were designed with CISC processors in mind. Many companies were unwilling to take a chance with the emerging RISC technology. Without commercial interest, processor developers were unable to manufacture RISC chips in large enough volumes to make their price competitive.

Another major setback was the presence of Intel. Although their CISC chips were becoming increasingly unwieldy and difficult to develop, Intel had the resources to plow through development and produce powerful processors. Although RISC chips might surpass Intel's efforts in specific areas, the differences were not great enough to persuade buyers to change technologies.

The Overall RISC Advantage

Today, the Intel x86 is arguable the only chip which retains CISC architecture. This is primarily due to advancements in other areas of computer technology. The price of RAM has decreased dramatically. In 1977, 1MB of DRAM cost about $5,000. By 1994, the same amount of memory cost only $6 (when adjusted for inflation). Compiler technology has also become more sophisticated, so that the RISC use of RAM and emphasis on software has become ideal.