Architettura del Calcolatore e Manipolazione dei Dati

Capitolo 2 del testo

Alberto Policriti

24 Ottobre, 2019

Il modello di von Neumann

4(5) Componenti Fondamentali

1. unità di elaborazione: CPU
2. memoria centrale: RAM
3. periferiche (memoria di massa)
4. bus di sistema

La Central Processing Unit: chi manipola davvero i dati

CPU
è l'unità centrale di elaborazione. Alloggiata sul microprocessore, dirige e controlla ogni attività del computer e coordina le attività di memoria e delle unità periferiche oltre ad eseguire tutte le operazioni aritmetiche e logiche relative ad esempio ad un programma che si sta eseguendo. Tutte le operazioni sono eseguite in bit (0 e 1).

La Central Processing Unit: chi manipola davvero i dati

Caratteristiche

- è elettronica
- interpreta le istruzioni del linguaggio macchina
- accede alle locazioni di memoria della RAM
- è costituita da molte componenti, delle quali la più importante è la Arithmetic Logic Unit (ALU)
- ... è il *processore*

La Central Processing Unit: chi manipola davvero i dati

La Central Processing Unit: chi manipola davvero i dati

Caratteristiche tecniche

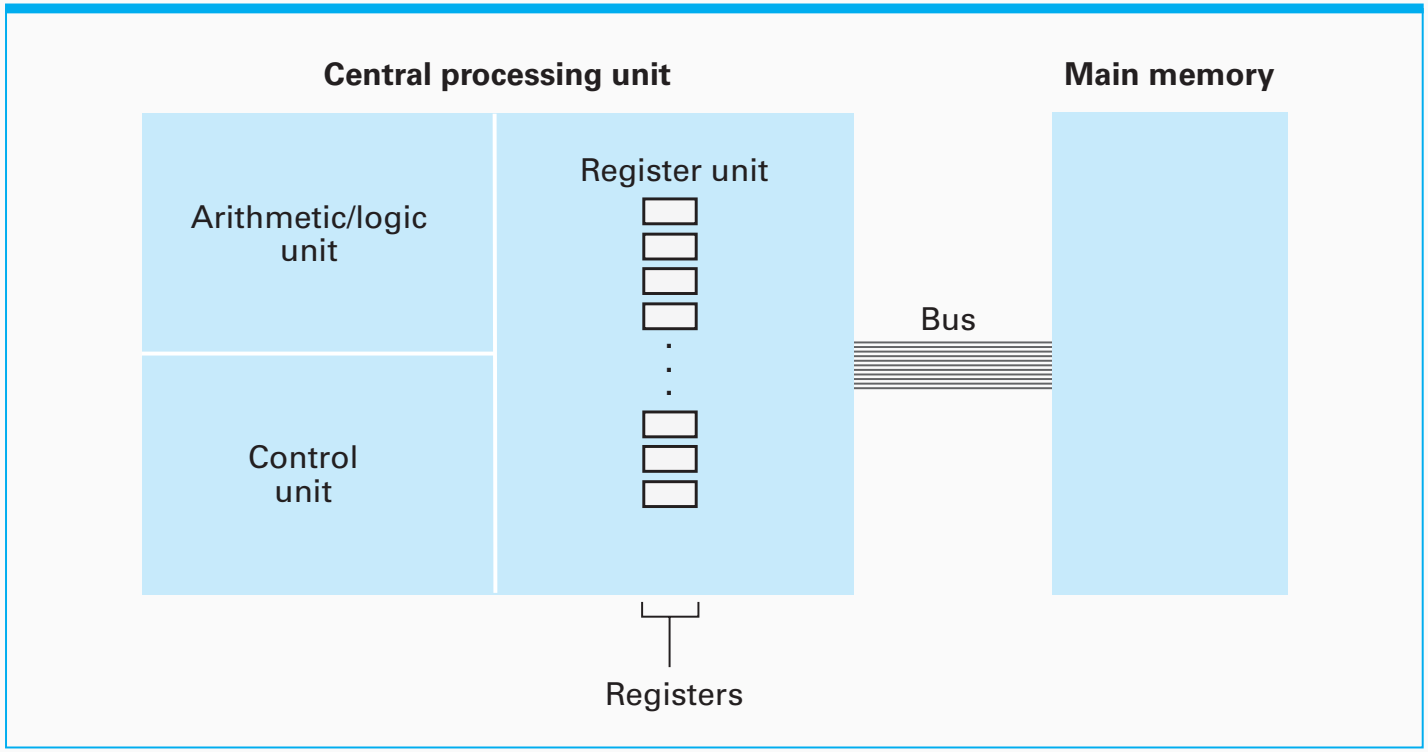
- Piccole (chiamate anche *microprocessori*)
- Montate su una *scheda madre*

Oggigiorno le CPU possono essere costituite da diversi "core", ognuno dei quali costituisce una CPU indipendente: di fatto si viene a creare un sistema "multiprocessore" utilizzando un solo integrato; Questo rende possibile l'esecuzione contemporanea di numerosi processi (programmi) senza rallentamento del sistema.

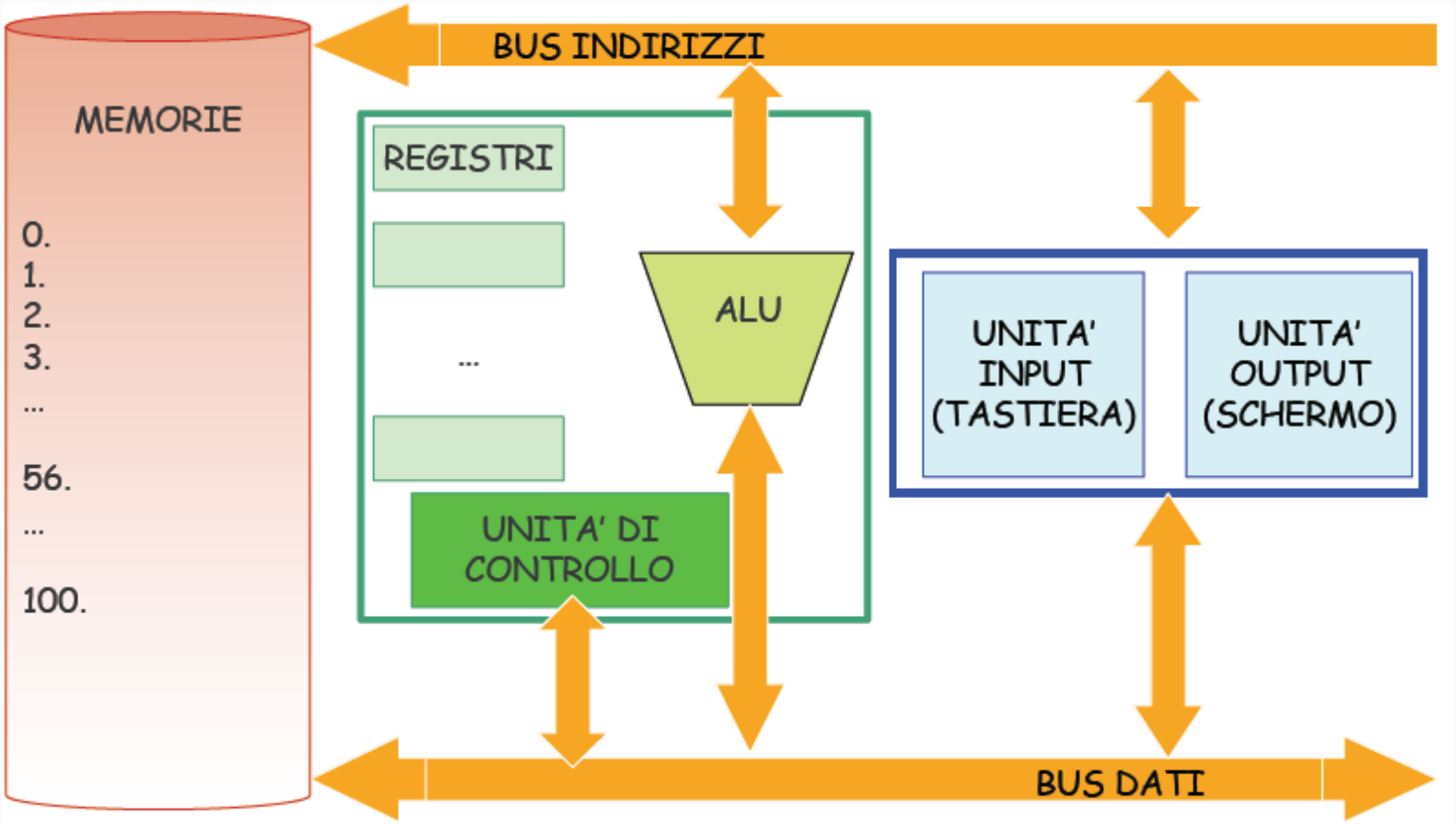
Come è fatta una CPU?

Tre componenti

1. Arithmetic-Logic Unit (ALU)
2. Control Unit
3. Register Unit (special/general purpose registers)



La visione d'insieme



Come si usa una CPU?

Cosa significa fare una somma?

1. Non solo l'algoritmo di somma
2. Si muovono dati
3. Si usano registri e indirizzi di memoria

Step 1. Get one of the values to be added from memory and place it in a register.

Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

Step 5. Stop.

Idee importanti

1. Stored-program (nella memoria principale)
2. Cache memory (e gerarchie di memorie)

Cache Memory

It is instructive to compare the memory facilities within a computer in relation to their functionality. Registers are used to hold the data immediately applicable to the operation at hand; main memory is used to hold data that will be needed in the near future; and mass storage is used to hold data that will likely not be needed in the immediate future. Many machines are designed with an additional memory level, called cache memory. **Cache memory** is a portion (perhaps several hundred KB) of high-speed memory located within the CPU itself. In this special memory area, the machine attempts to keep a copy of that portion of main memory that is of current interest. In this setting, data transfers that normally would be made between registers and main memory are made between registers and cache memory. Any changes made to cache memory are then transferred collectively to main memory at a more opportune time. The result is a CPU that can execute its machine cycle more rapidly because it is not delayed by main memory communication.

Idee importanti

Who Invented What?

Awarding a single individual credit for an invention is always a dubious undertaking. Thomas Edison is credited with inventing the incandescent lamp, but other researchers were developing similar lamps, and in a sense Edison was lucky to be the one to obtain the patent. The Wright brothers are credited with inventing the airplane, but they were competing with and benefited from the work of many contemporaries, all of whom were preempted to some degree by Leonardo da Vinci, who toyed with the idea of flying machines in the fifteenth century. Even Leonardo's designs were apparently based on earlier ideas. Of course, in these cases the designated inventor still has legitimate claims to the credit bestowed. In other cases, history seems to have awarded credit inappropriately—an example is the stored-program concept. Without a doubt, John von Neumann was a brilliant scientist who deserves credit for numerous contributions. But one of the contributions for which popular history has chosen to credit him, the stored-program concept, was apparently developed by researchers led by J. P. Eckert at the Moore School of Electrical Engineering at the University of Pennsylvania. John von Neumann was merely the first to publish work reporting the idea and thus computing lore has selected him as the inventor.

Collezione delle Istruzioni direttamente eseguibili dalla CPU
(Almeno) due filosofie diverse:

1. Reduced Instruction Set Computer (RISC: PowerPC - fixed length)
2. Complex Instruction Set Computer (CISC: Intel - variable length)

Comunque: poche istruzioni

Il Linguaggio Macchina

Tre gruppi:

1. data transfer (LOAD/STORE - I/O, ...)
2. arithmetic-logic (AND, OR, XOR, SHIFT, ROTATE, ...)
3. control (JUMP, STOP, ...)

Step 1. LOAD a register with a value from memory.

Step 2. LOAD another register with another value from memory.

Step 3. If this second value is zero, JUMP to Step 6.

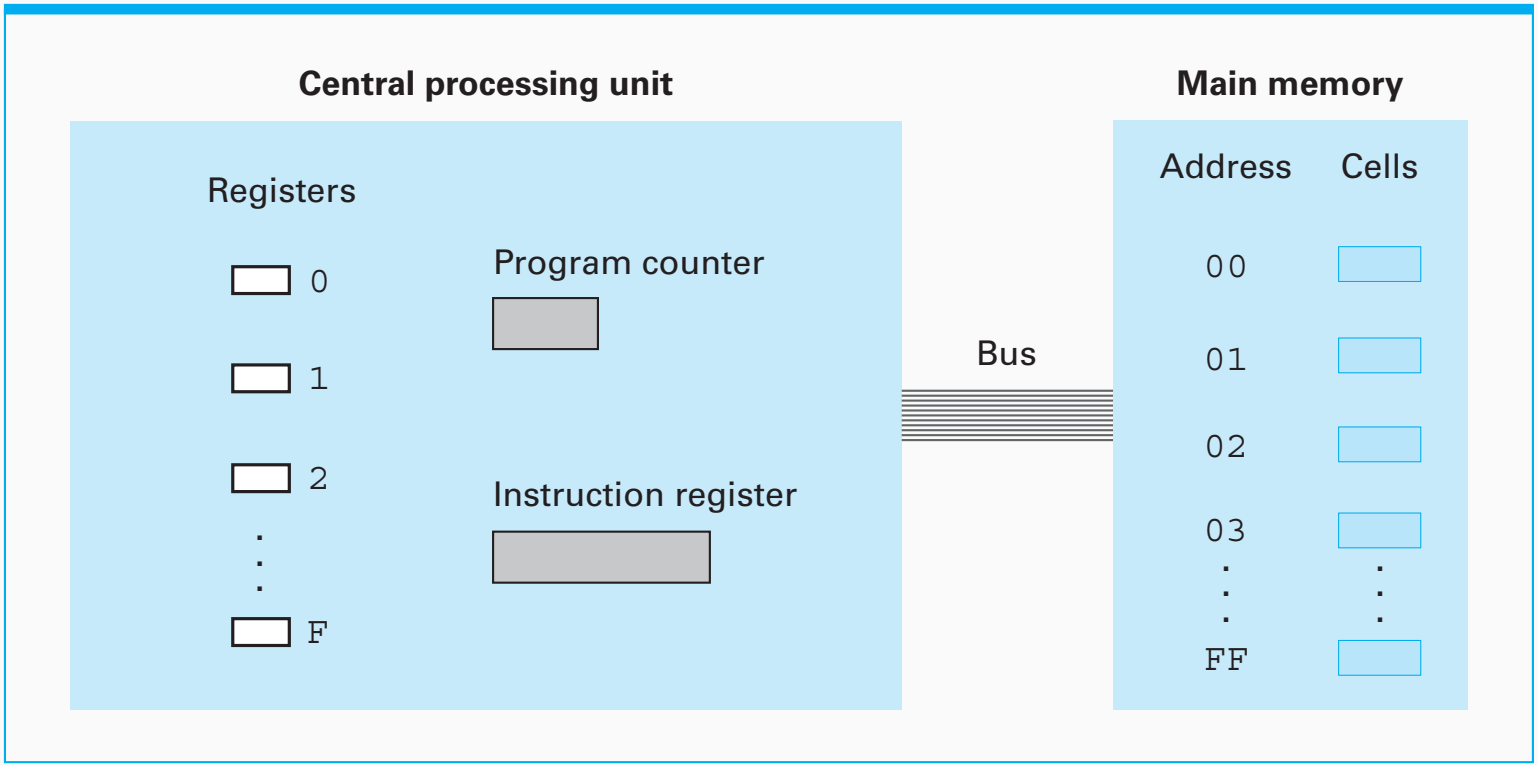
Step 4. Divide the contents of the first register by the second register and leave the result in a third register.

Step 5. STORE the contents of the third register in memory.

Step 6. STOP.

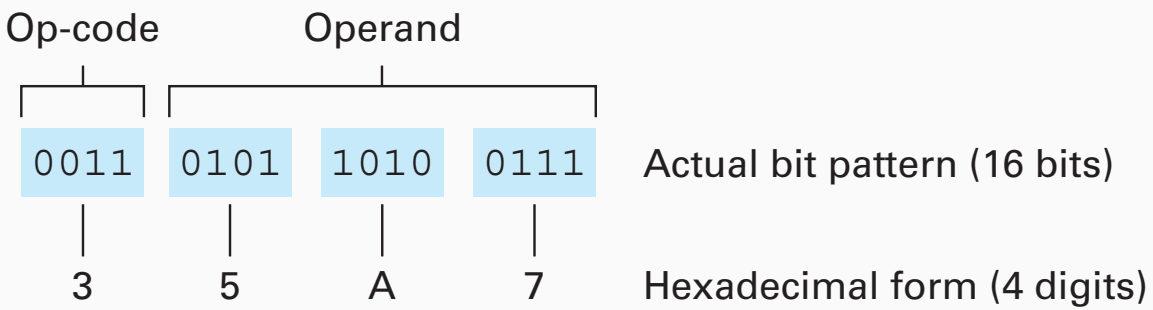
Esempio
Consideriamo una architettura da 16 registri e RAM da 256 celle.

Usiamo indirizzi scritti in esadecimale.



Codifica delle istruzioni macchina (Appendix C)

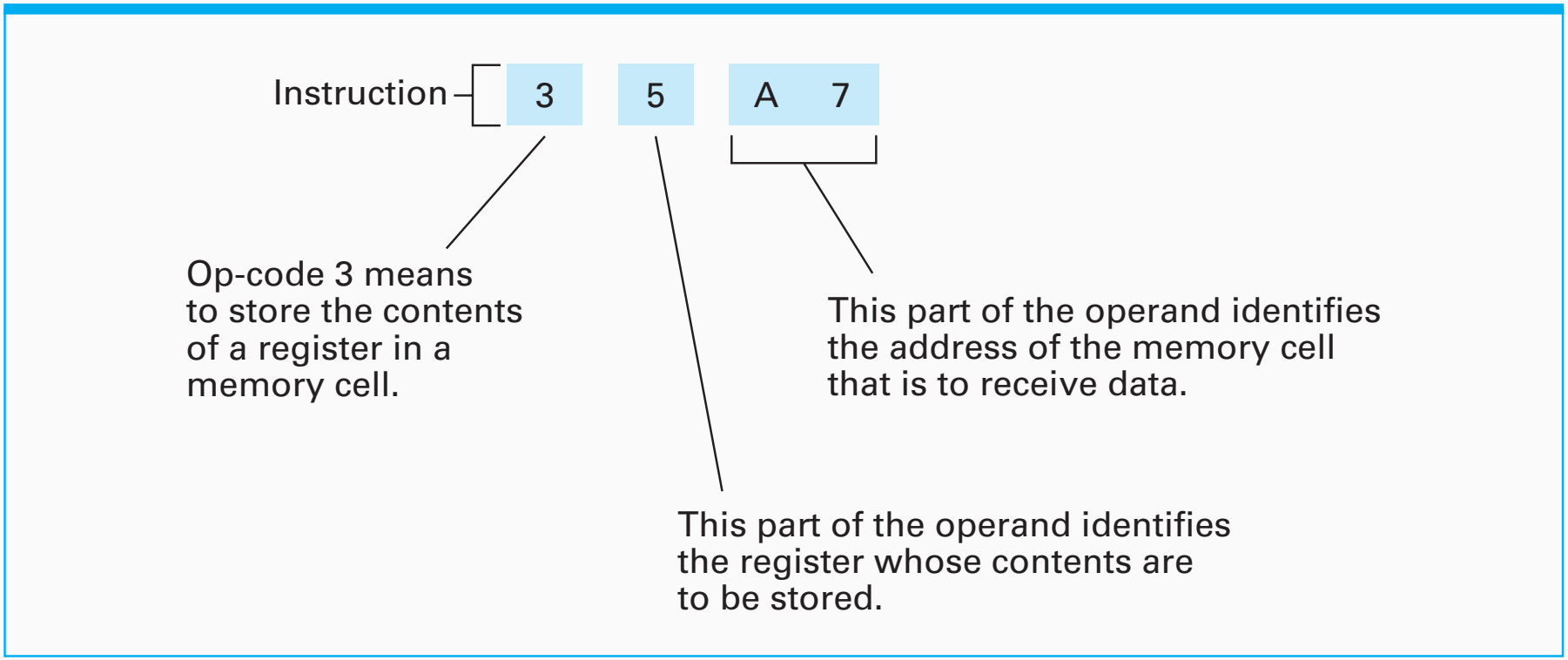
1. op-code
2. operand



N.B. Usiamo il primo carattere dell'operando per indicare il registro (ne abbiamo 16) e gli altri due per indicare la cella nella RAM (ne abbiamo 256). Potenze di 2.

35A7 in binario: 0011010110100111.

Il Linguaggio Macchina



Il Linguaggio Macchina

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

L'esecuzione di un programma

Le istruzioni devono venire:

1. Caricate (dalla memoria principale) nella CPU
2. Interpretate
3. Eseguite

Il ruolo dei registri

Due (importanti) registri *special purpose*

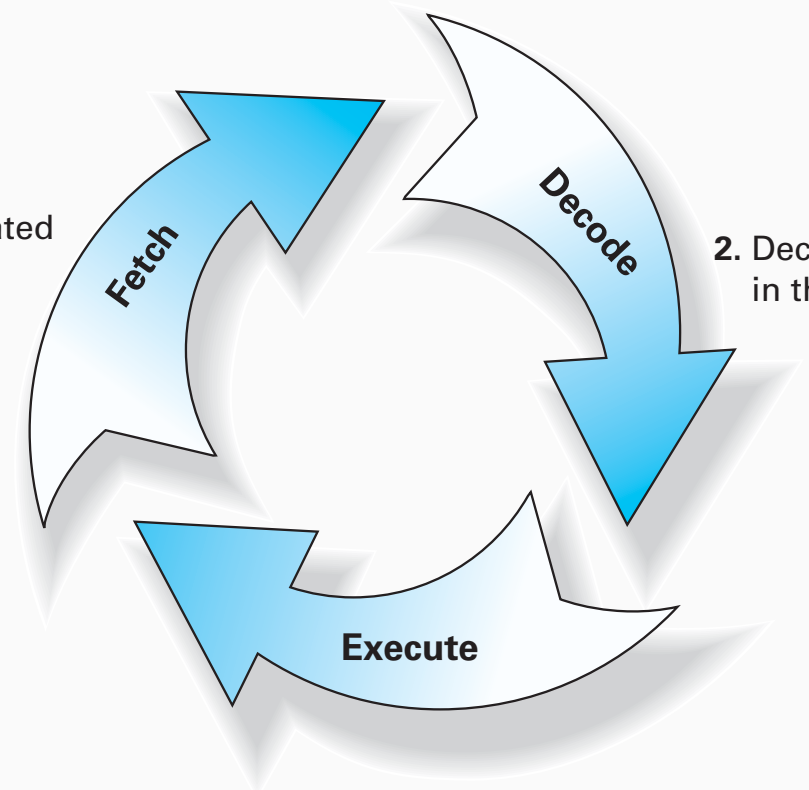
- instruction register
- program counter

Una importante nozione

- ciclo macchina

L'esecuzione di un programma

1. Retrieve the next instruction from memory (as indicated by the program counter) and then increment the program counter.



2. Decode the bit pattern in the instruction register.

3. Perform the action required by the instruction in the instruction register.

L'esecuzione di un programma

Domanda

Perché mi serve coordinare le attività della macchina?

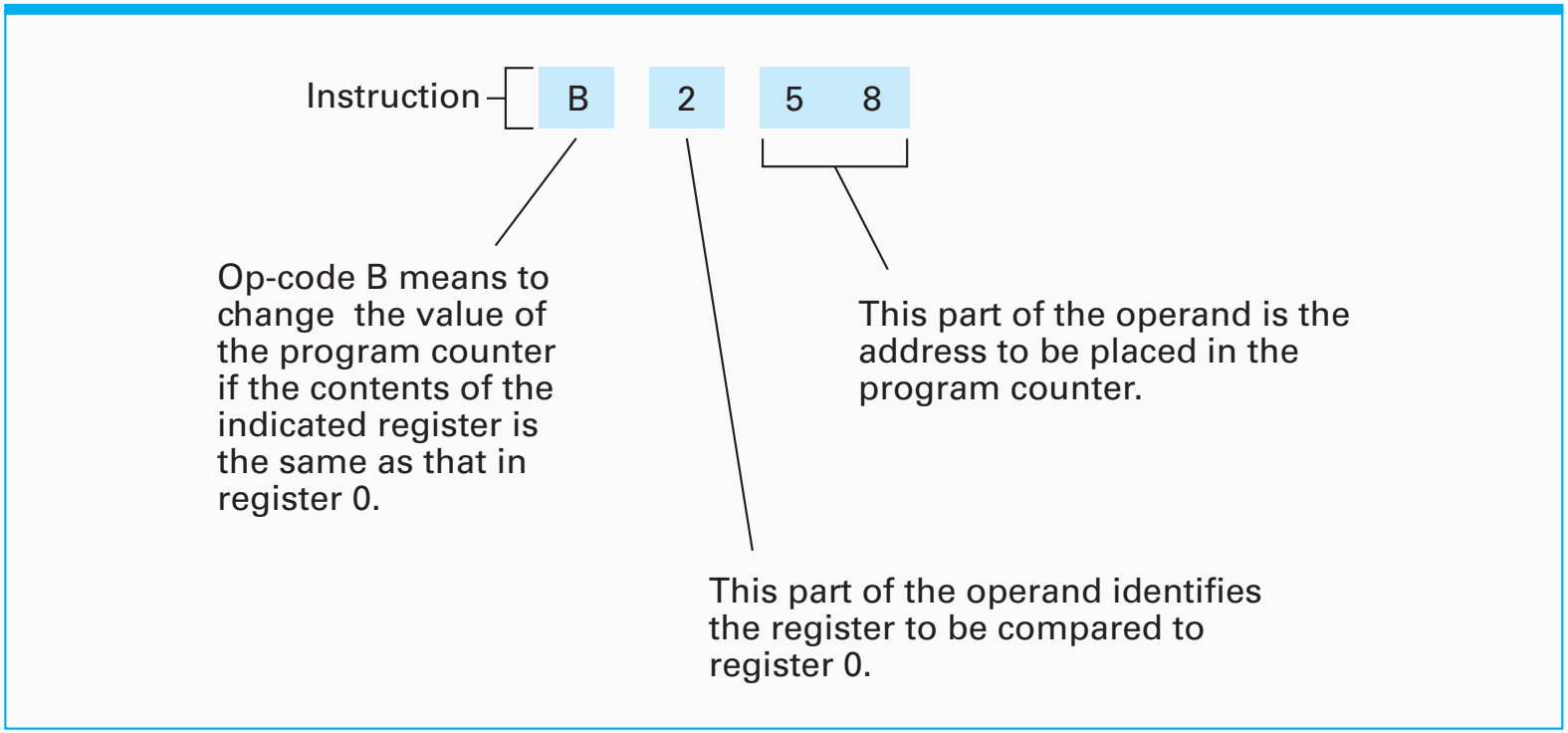
Perché mi serve un *clock*?

Comparing Computer Power

When shopping for a personal computer, you will find that clock speeds are often used to compare machines. A computer's **clock** is a circuit, called an oscillator, which generates pulses that are used to coordinate the machine's activities—the faster this oscillating circuit generates pulses, the faster the machine performs its machine cycle. Clock speeds are measured in hertz (abbreviated as Hz) with one Hz equal to one cycle (or pulse) per second. Typical clock speeds in desktop computers are in the range of a few hundred MHz (older models) to several GHz. (MHz is short for megahertz, which is a million Hz. GHz is short for gigahertz, which is 1000 MHz.)

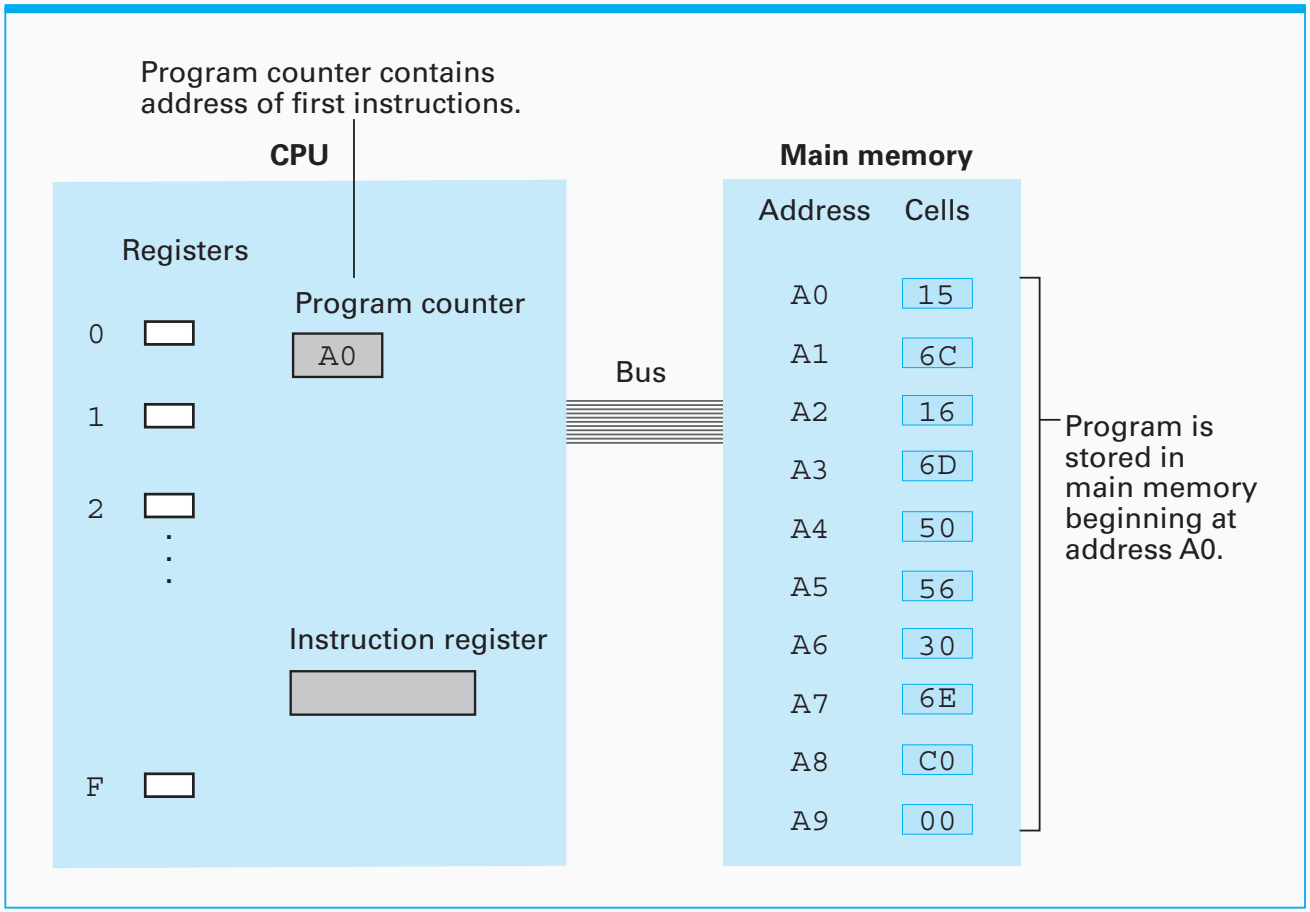
Unfortunately, different CPU designs might perform different amounts of work in one clock cycle, and thus clock speed alone fails to be relevant in comparing machines with different CPUs. If you are comparing a machine based on an Intel processor to one based on ARM, it would be more meaningful to compare performance by means of **benchmarking,** which is the process of comparing the performance of different machines when executing the same program, known as a benchmark. By selecting benchmarks representing different types of applications, you get meaningful comparisons for various market segments.

Una istruzione "particolare"



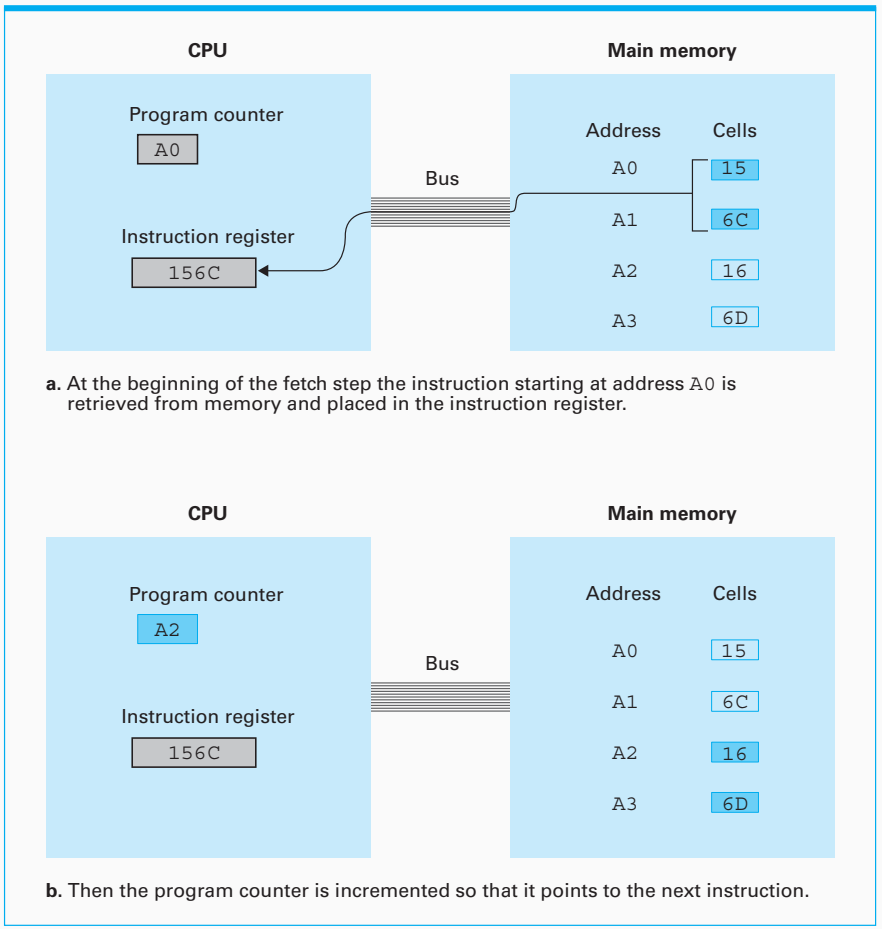
L'esecuzione di un programma

Esempio



L'esecuzione di un programma

Esempio



a. At the beginning of the fetch step the instruction starting at address A0 is retrieved from memory and placed in the instruction register.

b. Then the program counter is incremented so that it points to the next instruction.

Arithmetic/Logic operations

bitwise operations

```
  10011010
AND 11001001
  10001000
```

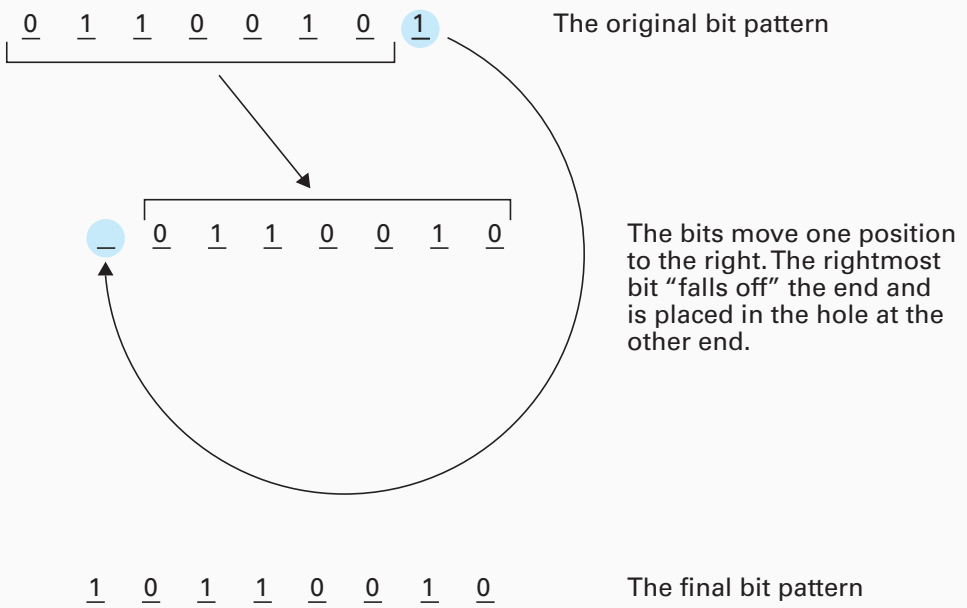
```
  10011010
OR 11001001
  11011011
```

```
   10011010
XOR 11001001
   01010011
```

mask bitmap shift ...

Arithmetic/Logic operations

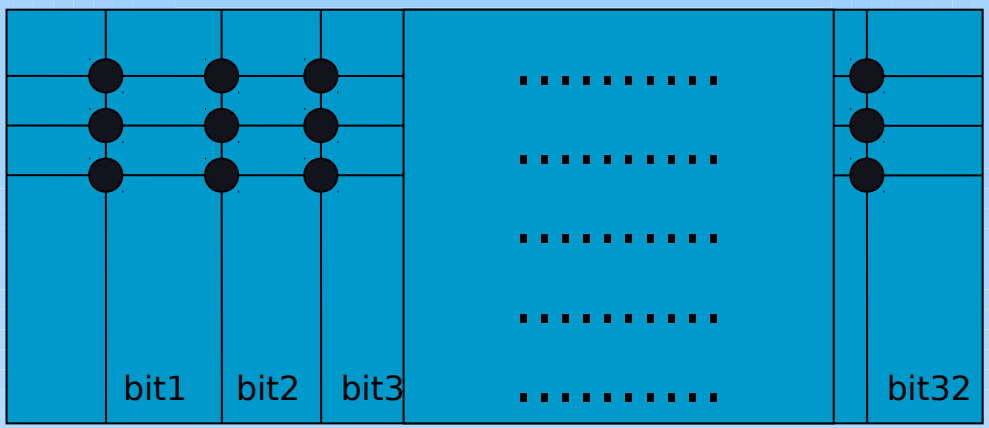
Figure 2.12 Rotating the bit pattern 65 (hexadecimal) one bit to the right



... arithmetic

Word

Parola (word) di memoria del calcolatore
La RAM è una sequenza di parole (word) che ... più sono e meglio è:



parole da 32 bit

Word

Question:
I've done some research. A byte is 8 bits and a word is the smallest unit that can be addressed on memory. The exact length of a word varies. What I don't understand is what's the point of having a byte? Why not say 8 bits?

I asked a prof this question and he said most machines these days are byte-addressable, but what would that make a word?

Word

Answer:
Byte: Today, a byte is almost always 8 bit. However, that wasn't always the case and there's no "standard" or something that dictates this. Since 8 bits is a convenient number to work with it became the de facto standard.

Word: The natural size with which a processor is handling data (the register size). The most common word sizes encountered today are 8, 16, 32 and 64 bits, but other sizes are possible. For examples, there were a few 36 bit machines, or even 12 bit machines.

Word

Answer:
The byte is the smallest addressable unit for a CPU. If you want to set/clear single bits, you first need to fetch the corresponding byte from memory, mess with the bits and then write the byte back to memory.

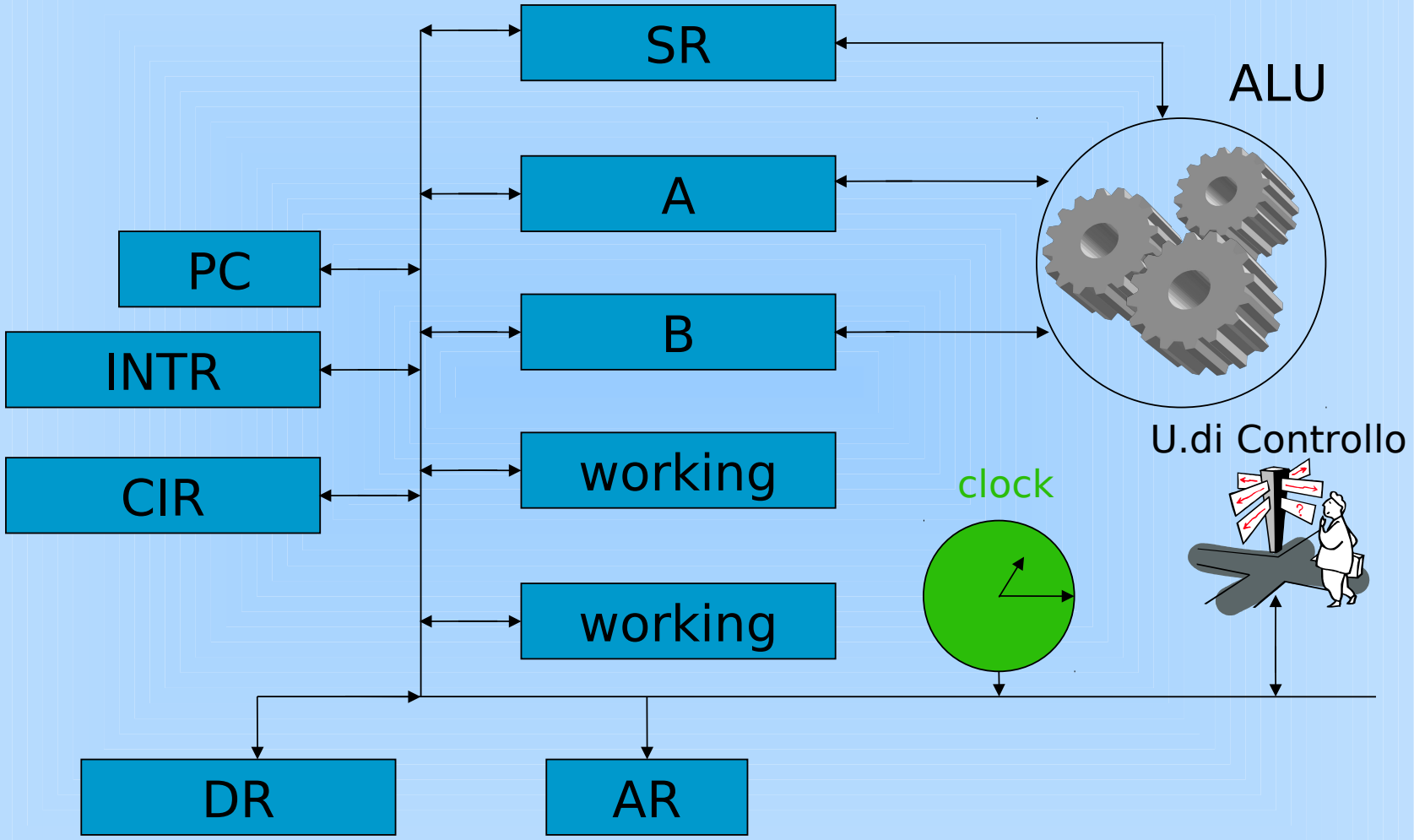
The word by contrast is biggest chunk of bits with which a processor can do processing (like addition and subtraction) at a time. That definition is to be take a bit loose, as some processor might have different word sizes for different tasks (integer vs. floating point processing for example). The word size is what the majority of operations work with.

There are also a few processors who have a different pointer size: for example, the 8086 is a 16-bit processor which means its registers are 16 bit wide. But its pointers (addresses) are 24 bit wide and were calculated by combining two 16 bit registers in a certain way.

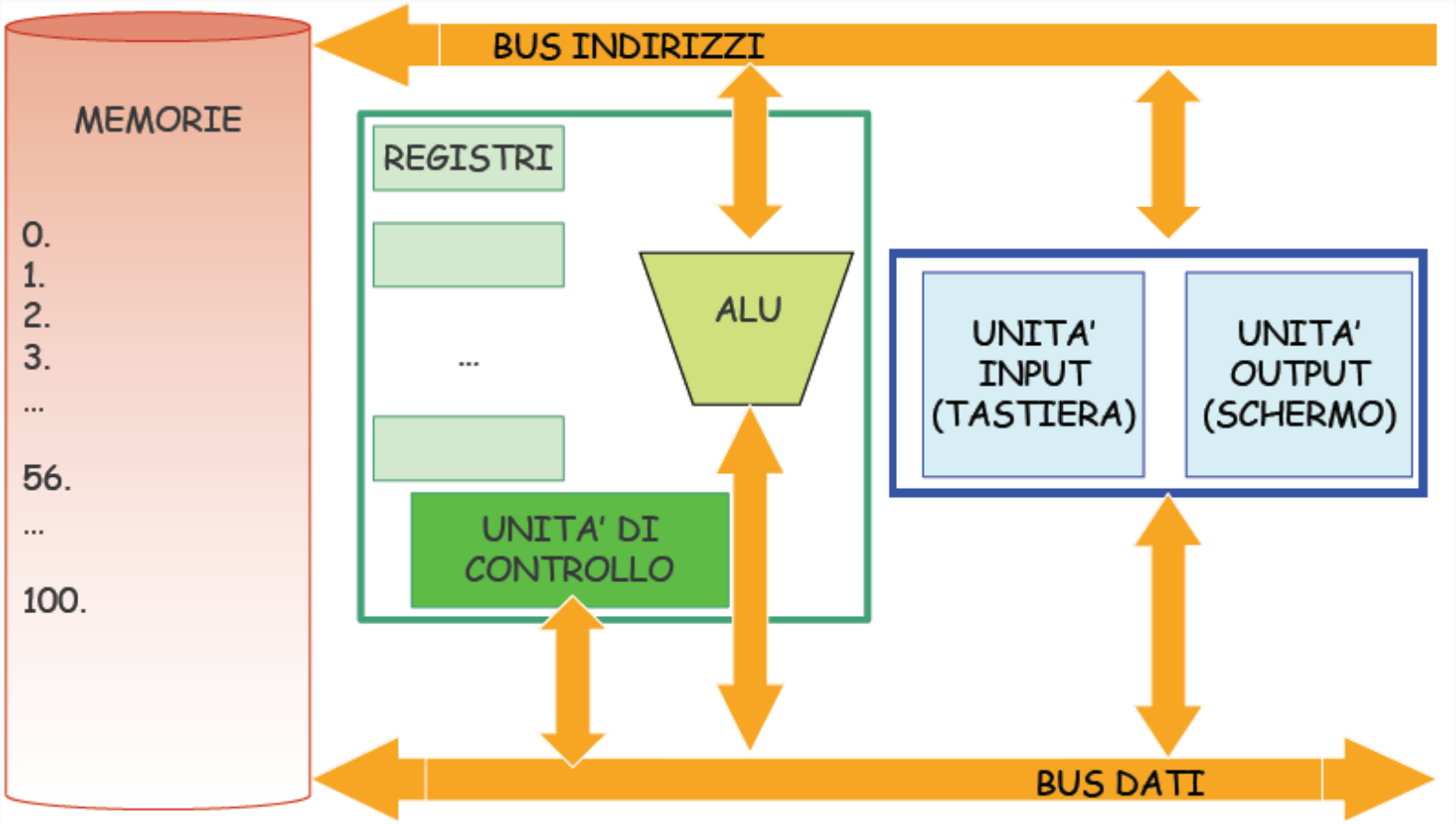
Componenti della CPU

- Unità di controllo
- Orologio di sistema
- Unità aritmetico logica (ALU)
- Registri (per l'accesso veloce)
- DR (Data Register, h bit), AR (Address Register, k bit)
- Registro Istruzione corrente (CIR) (Current Instruction Register, h bit)
- Program counter (PC) (k bit)
- Registo delle interruzioni (INTR)
- Registri operandi ALU: A e B
- Registri di lavoro e registro di stato (State Register SR)

Componenti della CPU



La visione d'insieme



Interfacce
Per comunicare con le periferiche si usano *"piccole CPU"* dedicate: interfacce (intelligenti).

Le interfacce sono dotate di unità di controllo proprie e di registri.

Registri per le interfacce di I/O

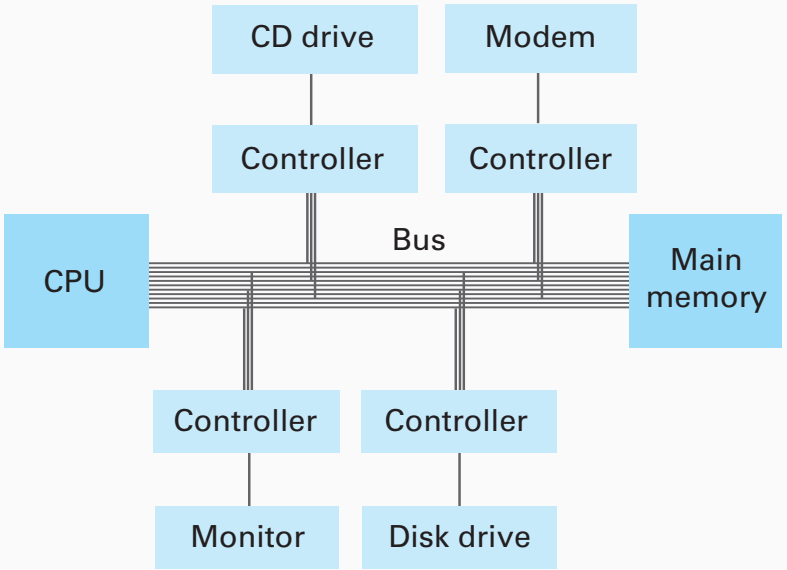
- Registro dati della periferica (PDR) manda i dati sul bus dei controlli
- Registro comando della periferica (PCR) manda i valori sul bus dei controlli
- Registro di stato della periferica (PSR) contiene informazioni sullo stato della periferica e può essere collegato direttamente al registro INTR

Controllers

A controller translates messages and data back and forth between forms compatible with the internal characteristics of the computer and those of the peripheral device to which it is attached. Originally, each controller was designed for a particular type of device; thus, purchasing a new peripheral device often required the purchase of a new controller as well.

Controllers

Figure 2.13 Controllers attached to a machine's bus



USB and FireWire

The universal serial bus (USB) and FireWire are standardized serial communication systems that simplify the process of adding new peripheral devices to a personal computer. USB was developed under the lead of Intel. The development of FireWire was led by Apple. In both cases the underlying theme is for a single controller to provide external ports at which a variety of peripheral devices can be attached. In this setting, the controller translates the internal signal characteristics of the computer to the appropriate USB or FireWire standard signals. In turn, each device connected to the controller converts its internal idiosyncrasies to the same USB or FireWire standard, allowing communication with the controller. The result is that attaching a new device to a PC does not require the insertion of a new controller. Instead, one merely plugs any USB compatible device into a USB port or a FireWire compatible device into a FireWire port.

Of the two, FireWire provides a faster transfer rate, but the lower cost of USB 2.0 technology has made it the leader in the lower-cost mass market arena. A new, faster version of the USB standard, version 3.0, has also begun to appear on the market. USB-compatible devices on the market today include mice, keyboards, printers, scanners, digital cameras, smartphones, and mass storage systems designed for backup applications. FireWire applications tend to focus on devices that require higher transfer rates such as video recorders and online mass storage systems.