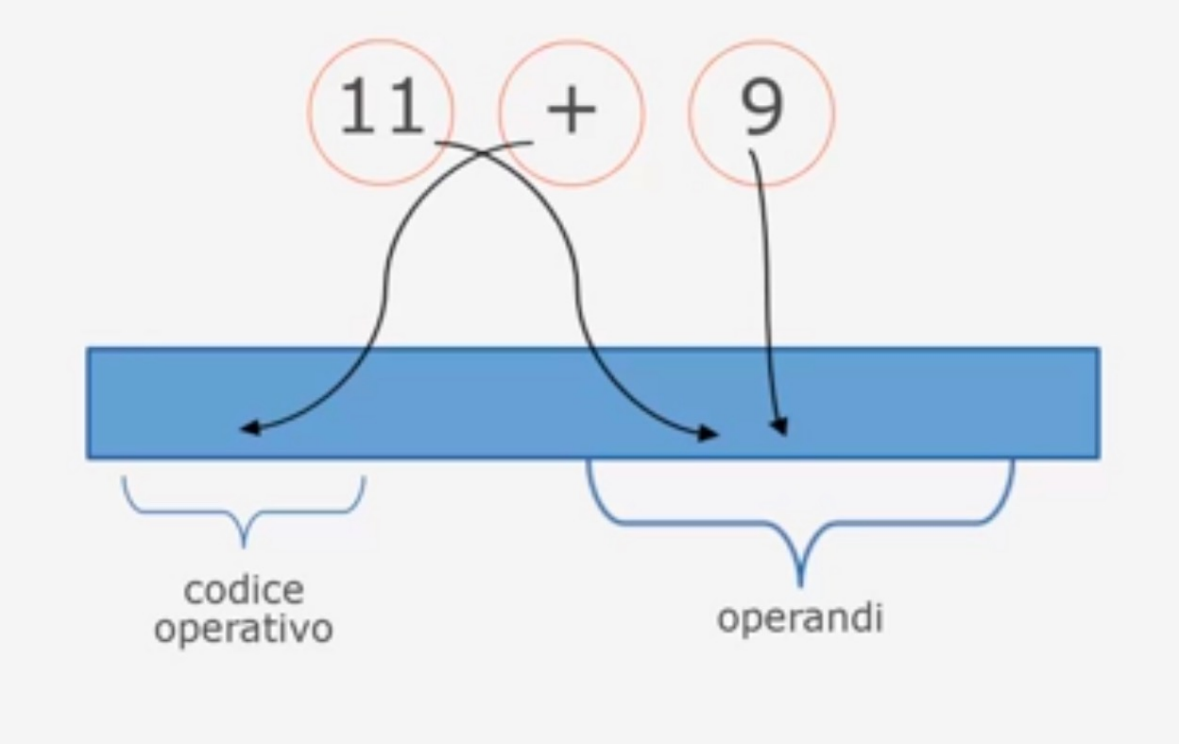
Proprio come codifichiamo testi e immagini, dobbiamo codificare le istruzioni dei programmi che vengono eseguiti da un computer.

Anche un programma è costituito da una sequenza di byte, suddivisi in *istruzioni*

Ogni istruzione è composta da un codice operativo, seguito da un certo numero di operandi.

Nel caso più semplice, il codice operativo e ogni operando occupano ciascuno esattamente un byte

Il codice operativo indica il tipo di operazione da compiere sugli operandi: somma, sottrazione, moltiplicazione, OR, AND, XOR, e così via



Possiamo usare l’analogia del cuoco per capire come il processore (o CPU) esegue i programmi

Per cucinare, il cuoco è in grado di eseguire un numero ben preciso di operazioni stabilite a priori.

Le operazioni sono descritte da un codice operativo, e utilizzano ingredienti e recipienti

Codice	Istruzione	Lun. (bytes)
`00 x y z`	versare y grammi di ingrediente x in recipiente z	4
`01 x y`	versare contenuto del recipiente x in recipiente y	3
`02 x`	spostare recipiente x sul fuoco	2
`03 x`	togliere recipiente x dal fuoco	2
`04 x`	spostare recipiente x sul lavandino	2
`05 x`	mescolare recipiente x	2
`06 x`	aspettare x minuti	2
`07 x`	mangiare/bere contenuto del recipiente x	2

Gli ingredienti e i recipienti su cui operano le istruzioni sono rappresentati da un ben preciso e univoco codice numerico:

Cod.	ingrediente
00	acqua
01	olio
02	sale
03	pasta
04	Passata di pomodoro
05	aglio

Cod.	recipiente
00	pentola
01	padella
02	scolapasta

codice operativo = 00
versare y grammi di ingrediente x in recipiente z

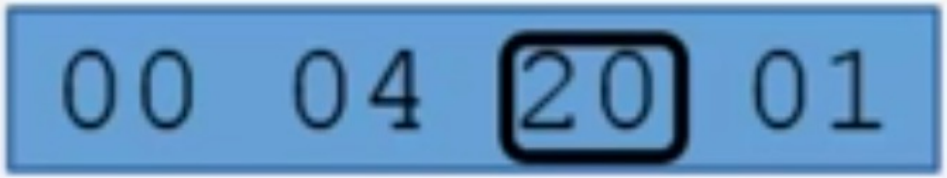
codice operativo = 00
versare y grammi di ingrediente x in recipiente z

Operando x = 04
Pomodoro

codice operativo = 00
versare y grammi di ingrediente x in recipiente z

Operando x = 04
Pomodoro

Operando y = 20 esadecimale (=32 decimale)
32 grammi di pomodoro



codice operativo = 00
versare y grammi di ingrediente x in recipiente z

Operando x = 04
Pomodoro

Operando y = 20 esadecimale (=32 decimale)
32 grammi di pomodoro

Operando z = 01
Padella

codice operativo = 00
versare y grammi di ingrediente x in recipiente z

Operando x = 04
Pomodoro

Operando y = 20 esadecimale (=32 decimale)
32 grammi di pomodoro

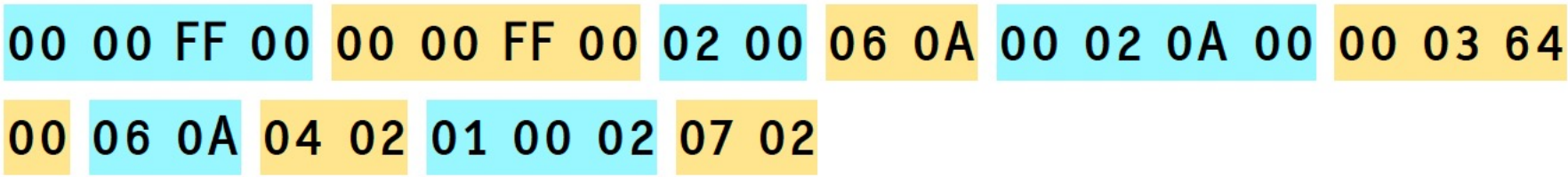
Operando z = 01
Padella

Istruzione codificata
"versare 32 grammi di pomodoro in padella"

00 04 20 01

La CPU preleva le istruzioni di un programma da eseguire una dopo l'altra dalla memoria principale (o dalla cache, come vedremo meglio più avanti).

Per ogni istruzione ne esamina il codice operativo ed esegue l'operazione corrispondente sugli operandi che seguono.



00 00 FF 00 00 00 FF 00 02 00 06 0A 00 02 0A 00 00 03 64 00 06 0A 04 02 01 00 02 07 02

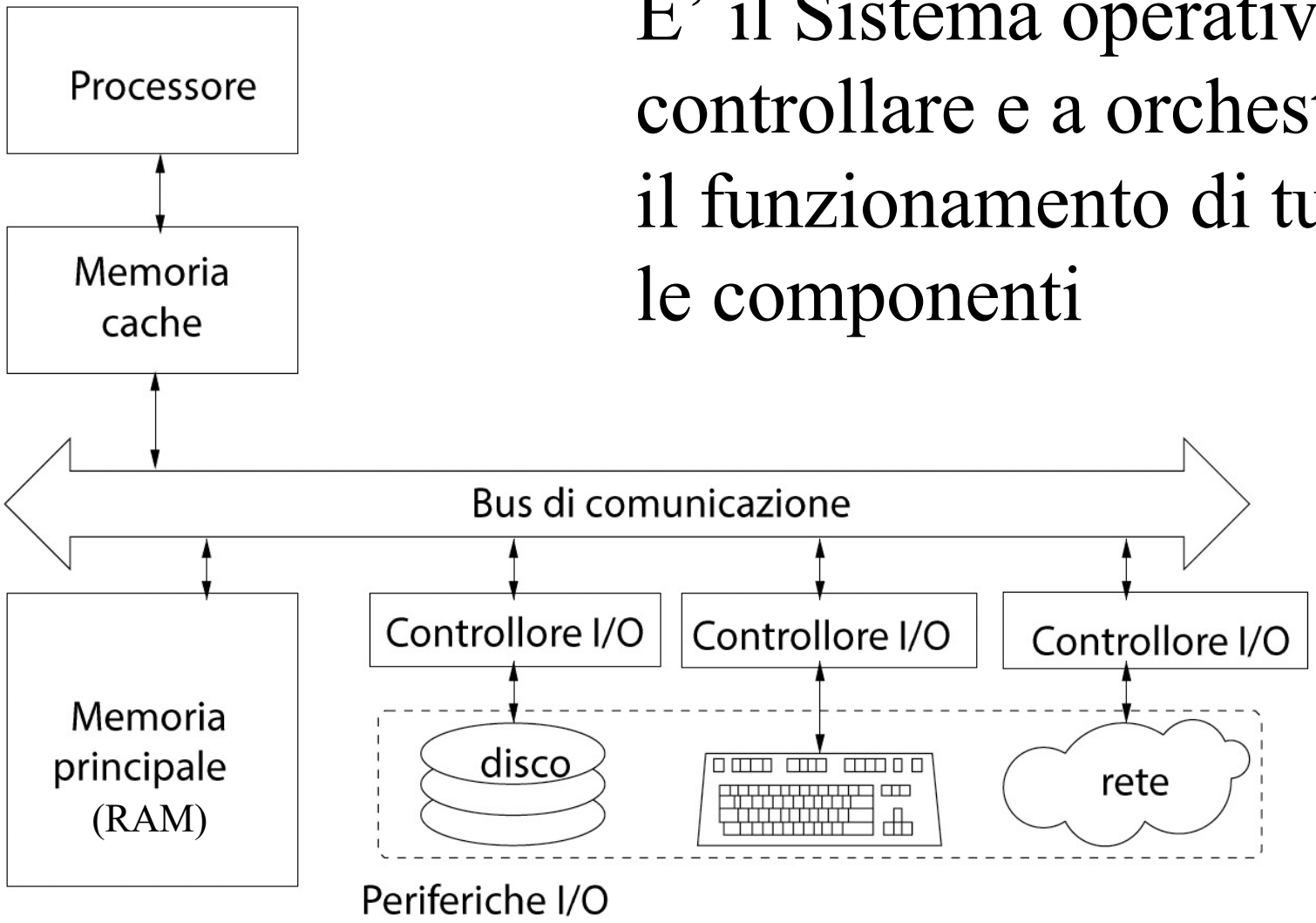
1. **00 00 FF 00**: l'istruzione con codice 00 indica l'istruzione "versare y grammi di ingrediente x in recipiente z". Andando a sostituire opportunamente gli operandi, scopriamo che l'istruzione corrisponde a "versare 255 g di acqua nella pentola"

2. **00 00 FF 00**: la seconda istruzione è identica

00 00 FF 00 **00 00 FF 00** **02 00** **06 0A** **00 02 0A 00** **00 03 64** **00** **06 0A** **04 02** **01 00 02** **07 02**

3. **02 00**: “spostare la pentola sul fuoco”
4. **06 0A**: “aspettare 10 minuti”, affinché l'acqua vada in ebollizione
5. **00 02 0A 00**: “versare 10 g di sale in pentola”
6. **00 03 64 00**: “versare 100 g di pasta nella pentola”
7. **06 0A**: “aspettare 10 minuti”, per la cottura della pasta
8. **04 02**: “spostare lo scolapasta sul lavandino”
9. **01 00 02**: “versare contenuto della pentola nello scolapasta”
10. **07 02**: “mangiare il contenuto dello scolapasta

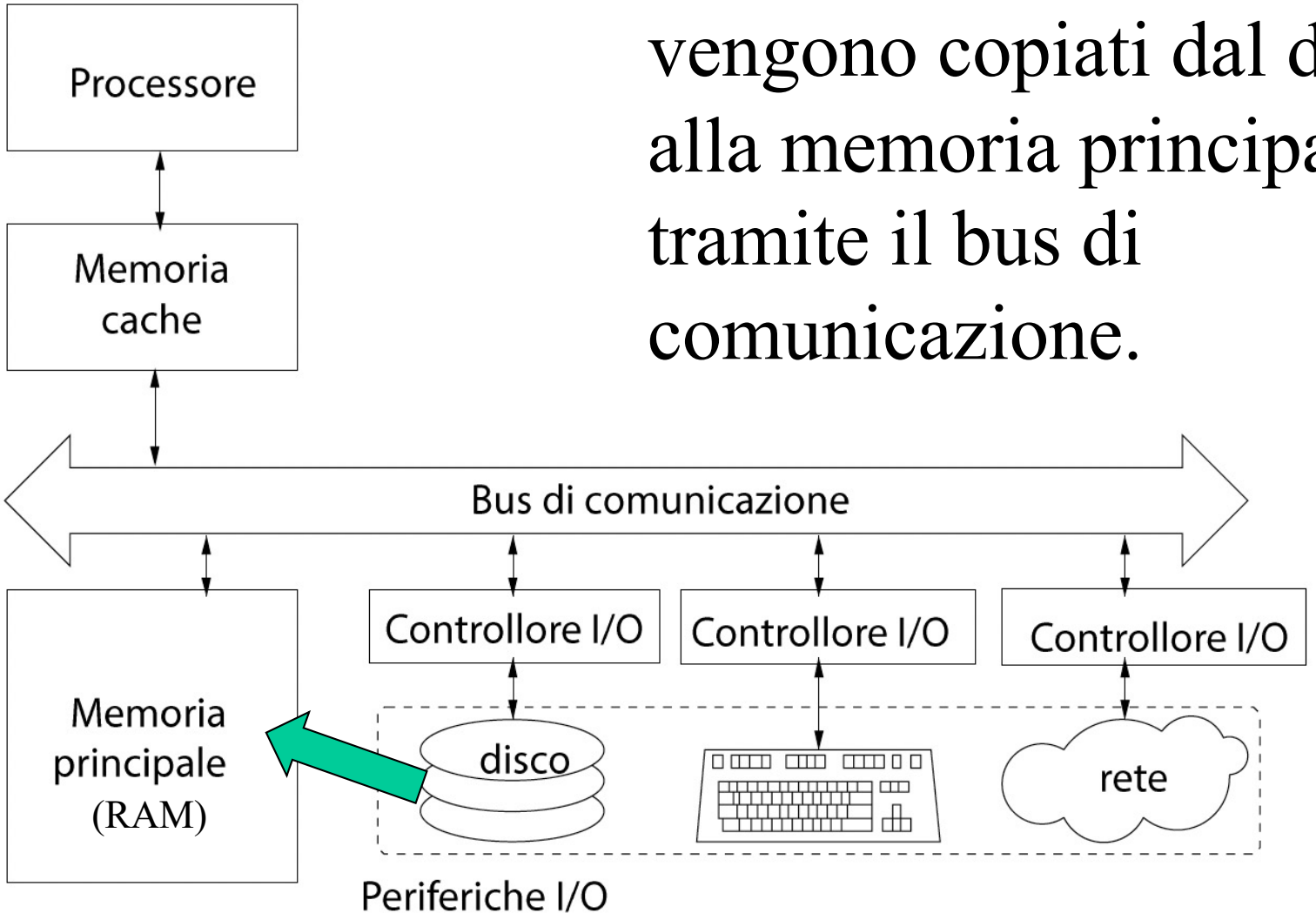
Componenti di un computer:



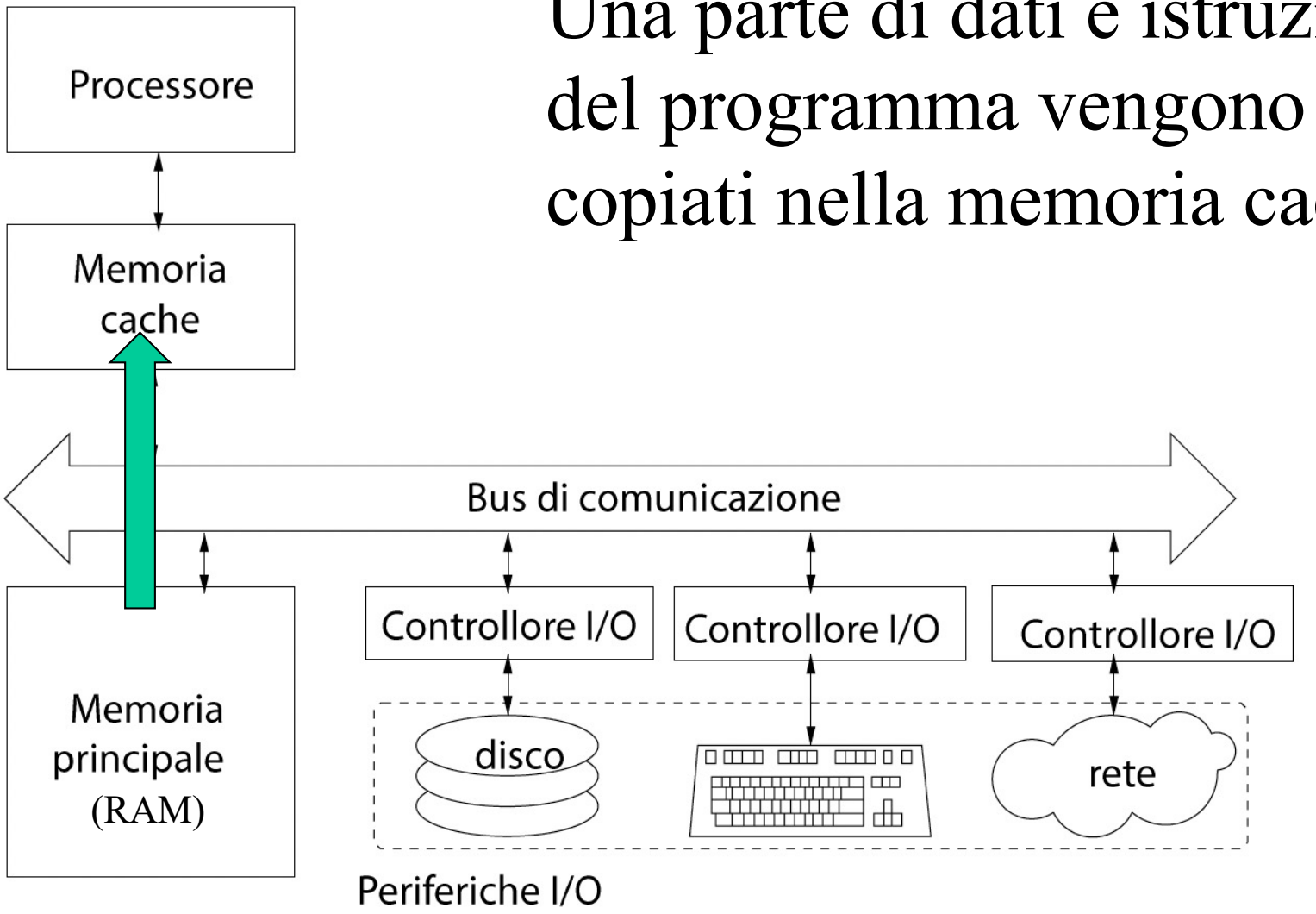
E' il Sistema operativo a controllare e a orchestrare il funzionamento di tutte le componenti

Quando lanciamo un programma/applicazione:

Il programma e i dati vengono copiati dal disco alla memoria principale tramite il bus di comunicazione.

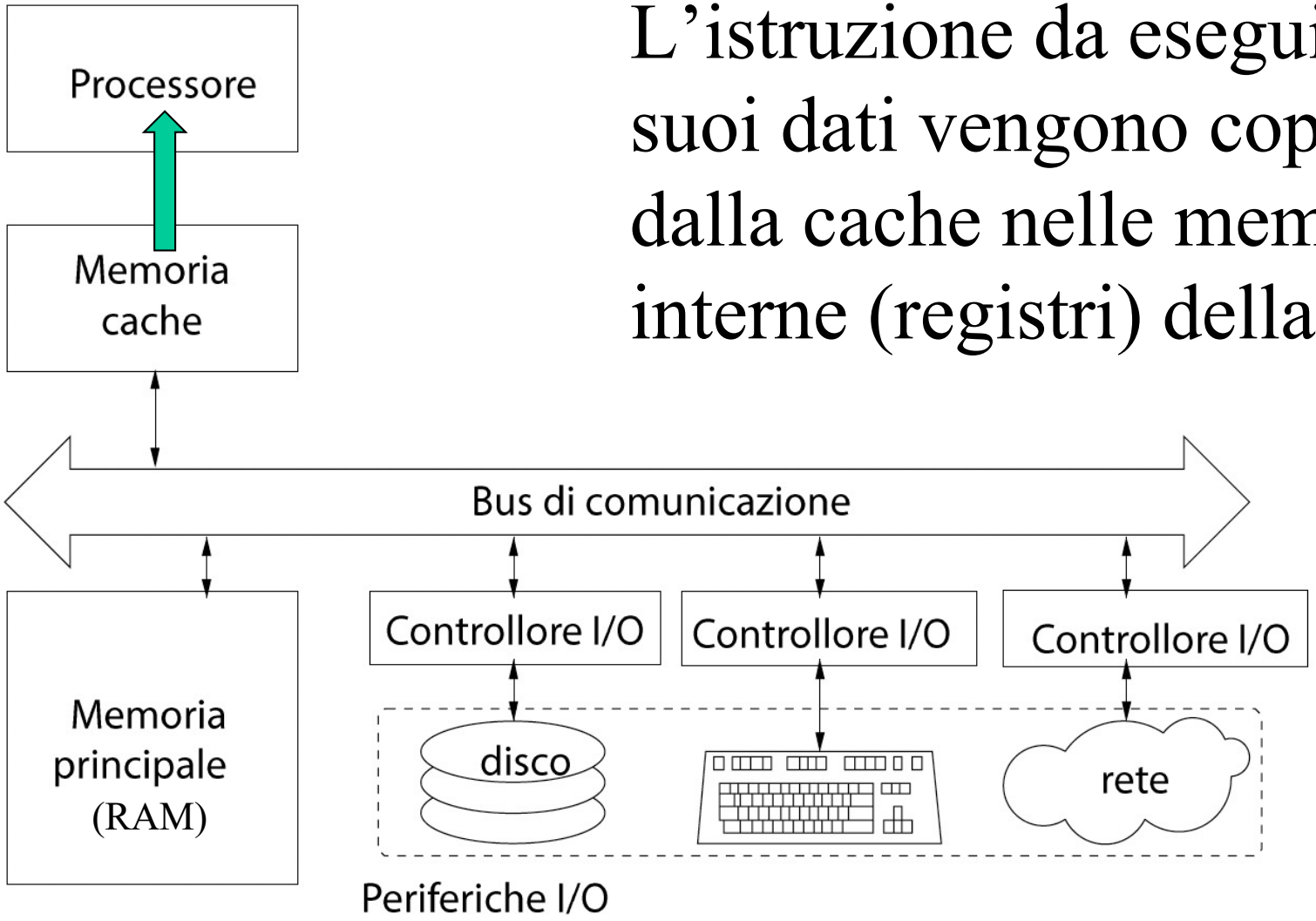


Durante l'esecuzione del programma:



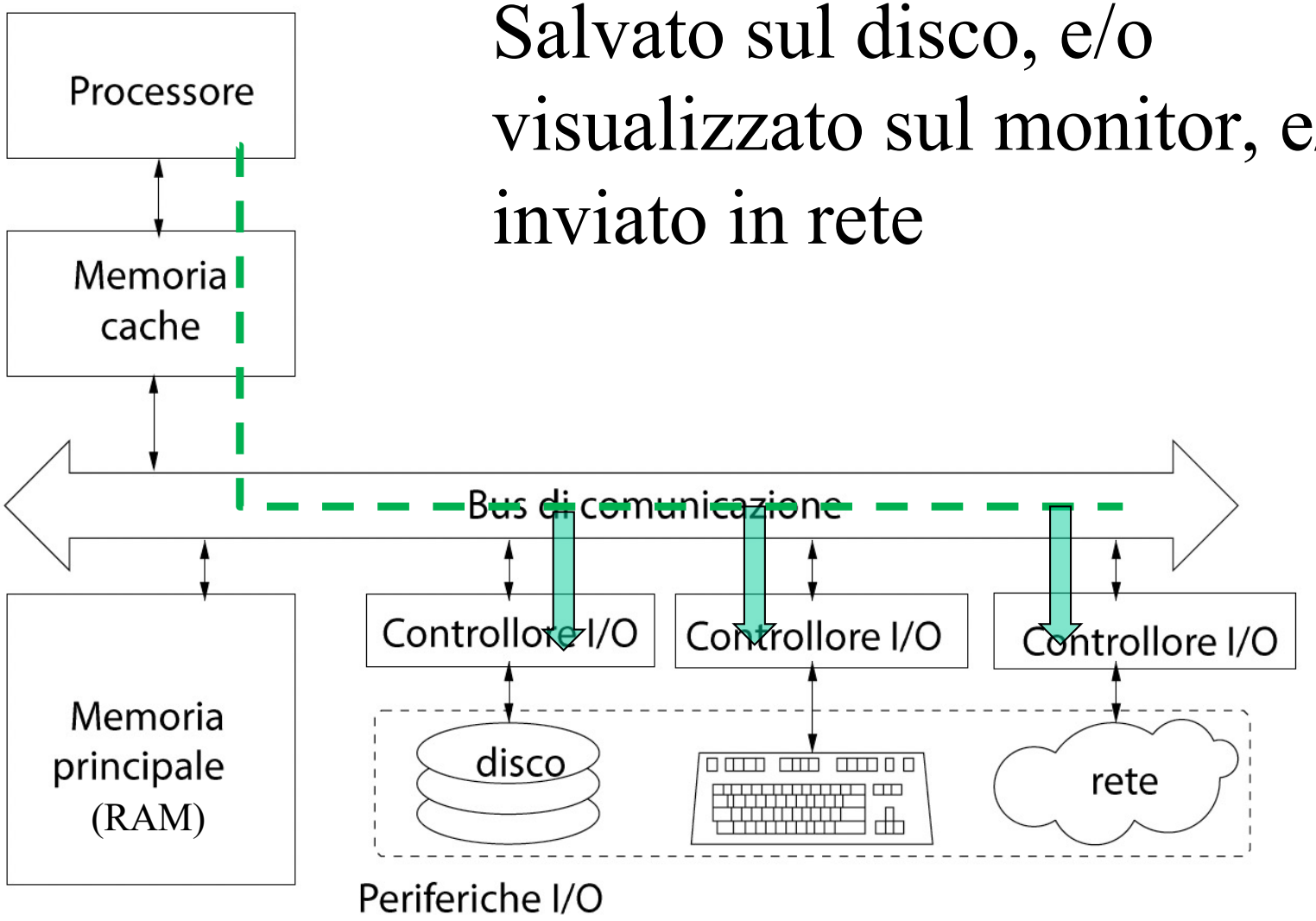
Una parte di dati e istruzioni del programma vengono copiati nella memoria cache

Per eseguire ogni singola istruzione la CPU accede alla memoria cache:



L'istruzione da eseguire e i suoi dati vengono copiati dalla cache nelle memorie interne (registri) della CPU

Al termine del programma, il suo output sarà:

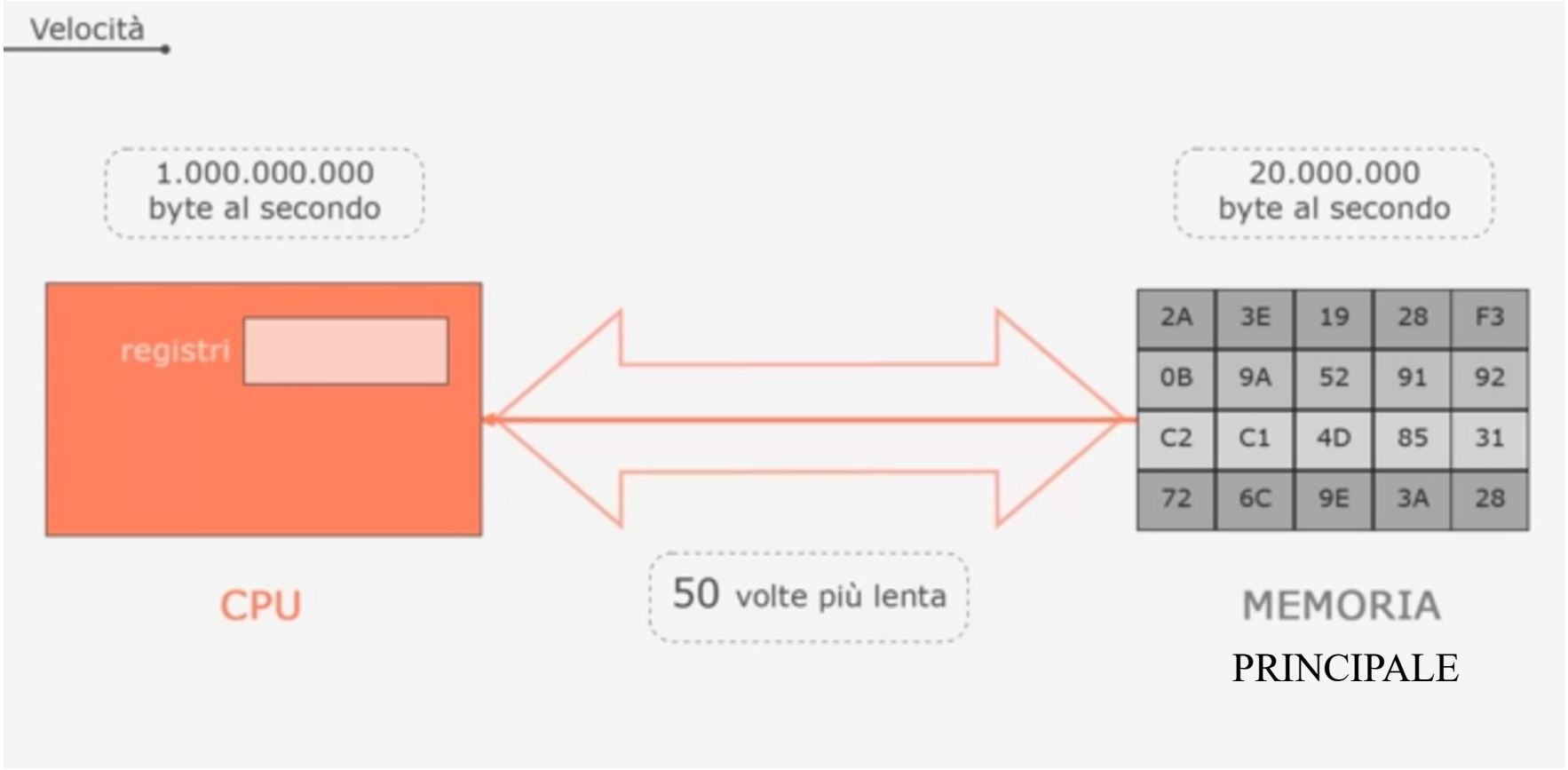


Salvato sul disco, e/o visualizzato sul monitor, e/o inviato in rete

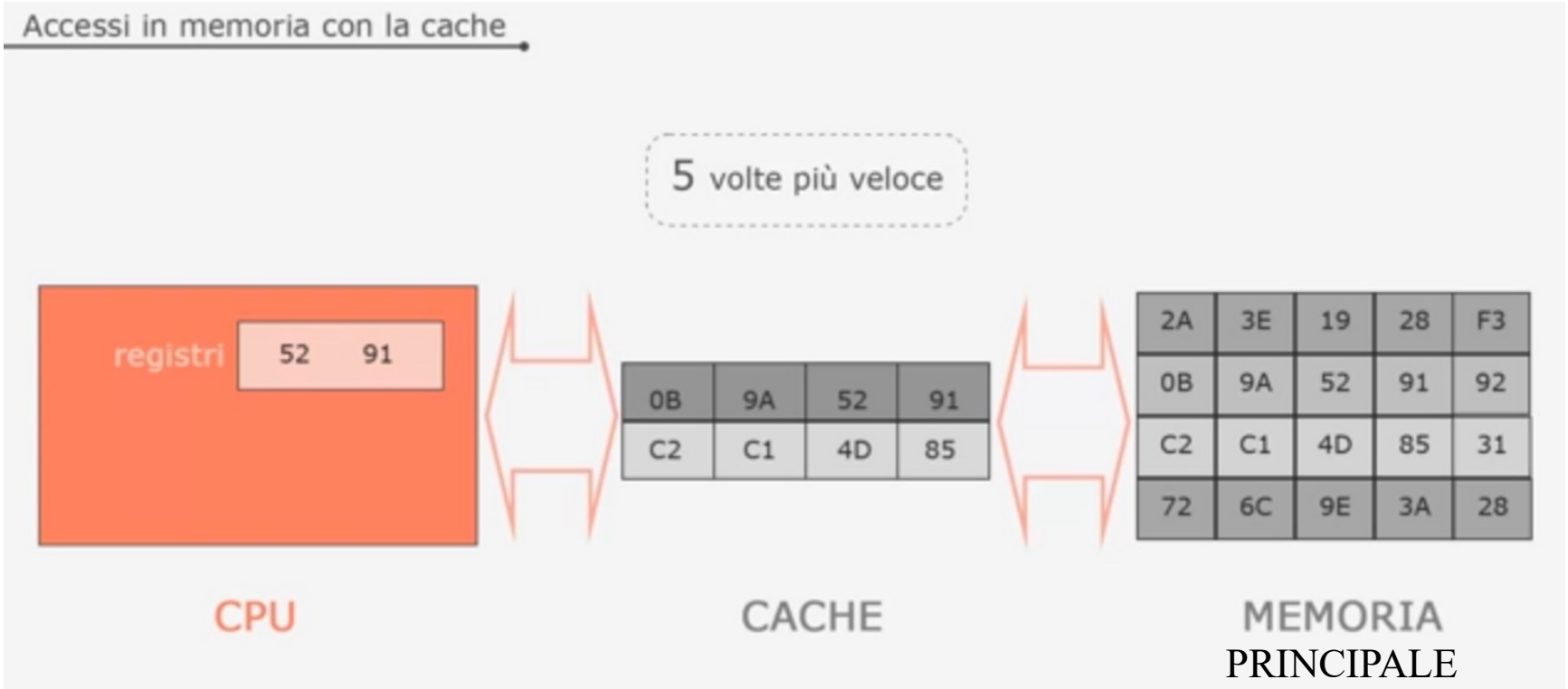
Perché abbiamo bisogno di tutte queste memorie?

Perché lavorano a velocità diverse

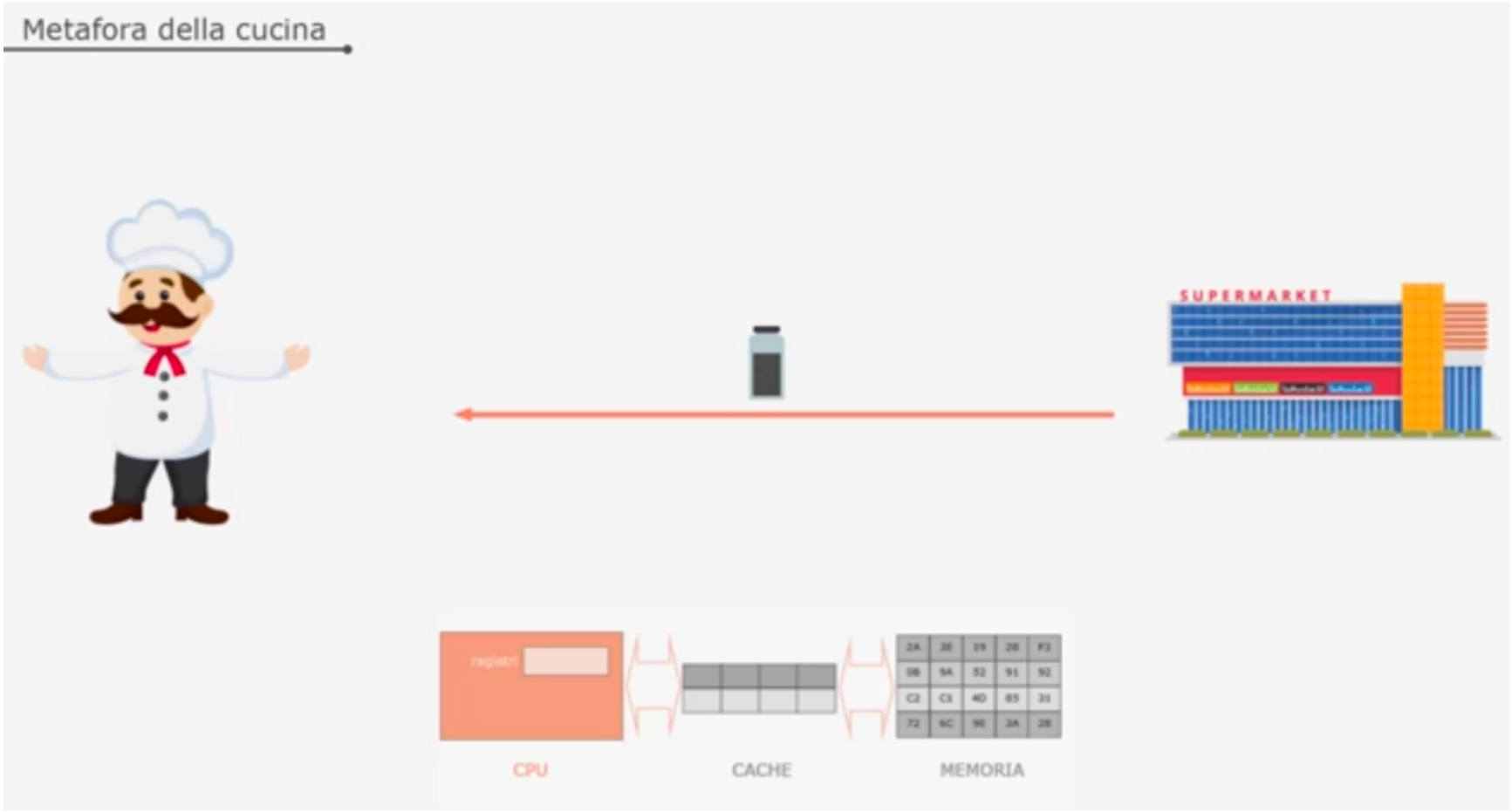
La memoria principale è molto più lenta della CPU



la cache è molto più veloce della memoria principale, e se ci copiamo dentro istruzioni e dati che ci servono subito, la CPU riesce a lavorare più velocemente

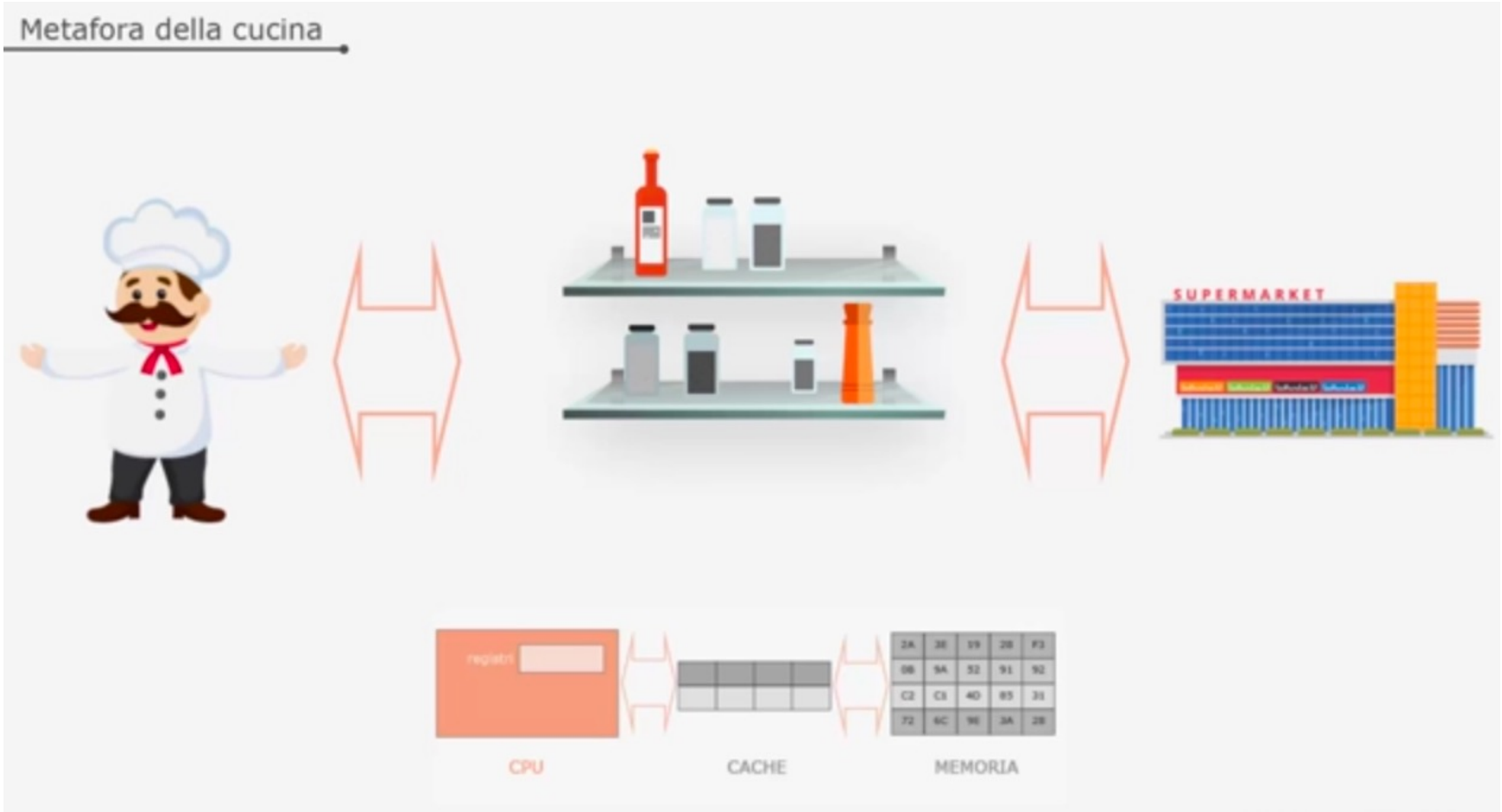


Nella metafora della cucina, **la memoria principale corrisponde al supermercato**: se il cuoco dovesse andare fino a lì per prendere ogni ingrediente che gli serve perderebbe un sacco di tempo

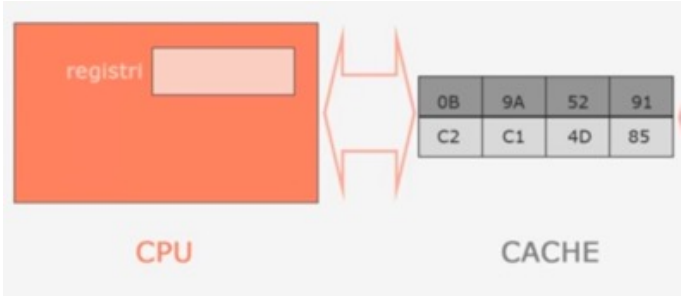


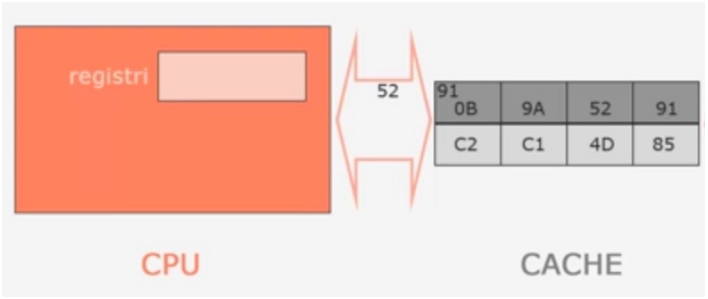
Quando il cuoco va al supermercato, preleva più ingredienti, che gli serviranno nell'immediato futuro, in modo da riempire **la sua dispensa, ossia la cache.**

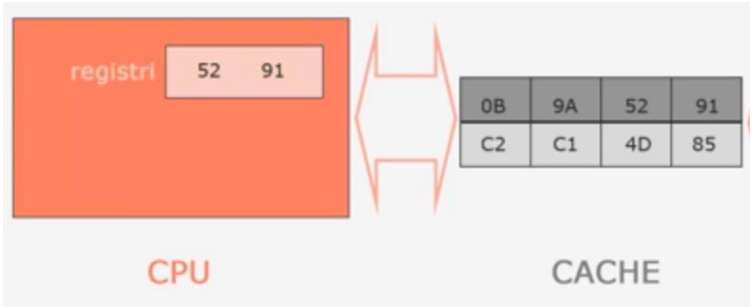
Così, potrà eseguire più velocemente le sue ricette.



Il processore accede alla cache per prelevare ogni singola istruzione e i suoi dati, e li copia in piccole memorie interne alla CPU: **i registri, che corrispondono al piano di lavoro del cuoco**







Perché abbiamo bisogno dei diversi livelli di memoria?

•Il disco è molto capiente ma lentissimo (possiamo vederlo come il grossista che rifornisce il supermercato)

•La memoria principale è più veloce del disco, ma può contenere meno dati e istruzioni del disco

•La cache è più veloce della memoria principale, ma può contenere pochi dati e istruzioni

•Il registri della CPU sono più veloci della cache, ma possono solo contenere l'istruzione in esecuzione e i dati che usa.

Il disco ci serve perché così le informazioni che contiene si conservano quando spegniamo il PC.

Ma perché non usiamo una cache molto più grande al posto della memoria principale?

Perché la cache è costruita con una tecnologia che
la rende si più veloce della memoria principale, ma anche più ingombrante e più costosa, quindi possiamo permettercene di meno.

Cosa analoga vale per i registri del processore nei confronti della cache.

Registri: centinaia di byte (< di un n.sec.)

Cache: pochi milioni di byte (10 n.sec)

Mem. principale: decine di miliardi di byte (50 n.sec)

SSD: centinaia di miliardi di byte (microsecondi)

Disco: migliaia di miliardi di byte (millisecondi)

Tipo di memoria	access time (nsec)	cap. (B)	Costo (€/B)
Registri	1	100	(difficile) 0.1
Cache	10	10^7	(diff.) 10^{-5}
Main memory	50	4×10^9	10^{-8}
disco SSD	10^5	10^{11}	5×10^{-10}
disco HDD	10^7	10^{11}	3×10^{-10}
USB key	10^7	10^{10}	5×10^{-10}
cloud storage	(diff.) 10^8	10^{12}	10^{-10}/anno