Architettura del computer

L'**architettura del computer** di von Neumann prevede due componenti fondamentali tra loro interconnesse:

- l'**unità centrale di elaborazione** denominata **CPU** (Central Processing Unit);
- la cosiddetta **memoria ad accesso casuale**. (**RAM**, Random Access Memory);

La **CPU** è a sua volta composta:

- dall'**unità di controllo** (**CU**, *Control Unit*), che sovrintende al funzionamento della macchina (in particolare al controllo della sequenza delle istruzioni da eseguire);
- dall'**unità aritmetico-logica** (**ALU**, *Arithmetic-Logic Unit*), all'interno della quale si svolgono le operazioni specificate dalle istruzioni.

La memoria **RAM** è organizzata come una sequenza di locazioni – identificate da **indirizzi** consecutivi – ciascuna delle quali può contenere un numero intero che codifica un dato o un'istruzione.

La struttura di interconnessione tra CPU e RAM è costituita da un *bus* unidirezionale per gli indirizzi e da un *bus* bidirezionale per i dati e le istruzioni

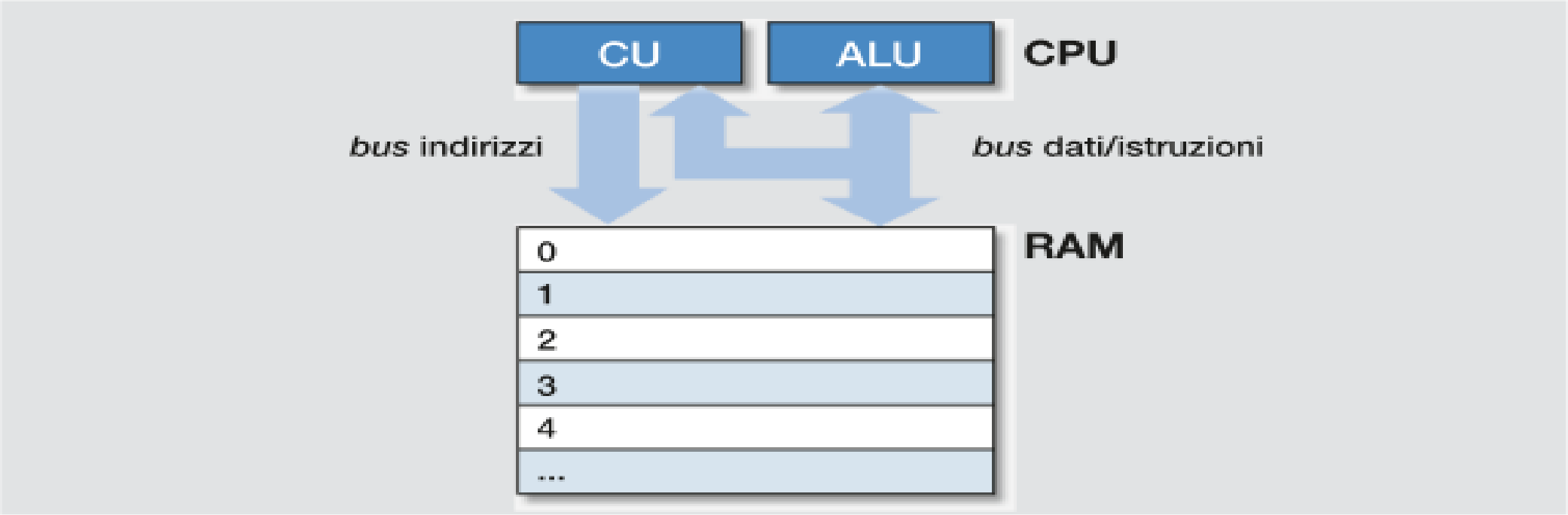


FIGURA 1

L'unità di controllo può richiedere alla memoria **operazioni di lettura** (trasferimento di dati o istruzioni dalla RAM alla CPU) o di **scrittura** (trasferimento di dati dalla CPU alla RAM).

L'operazione richiesta coinvolge la locazione di memoria indirizzata dal numero inviato alla RAM mediante il bus indirizzi, mentre i dati (o le istruzioni) transitano da un'unità all'altra attraverso il bus dati/istruzioni.

OSSERVAZIONE L'idea fondamentale che sta alla base della macchina ideata da von Neumann più di mezzo secolo fa e su cui ancora oggi è fondata l'architettura dei computer è la coesistenza in un'unica unità di memoria sia dei dati da utilizzare per il calcolo, sia dei risultati intermedi e finali prodotti dall'esecuzione del calcolo, sia delle istruzioni che specificano i calcoli da effettuare sui dati memorizzati.

Tutte queste «informazioni» sono memorizzate nella memoria della macchina in formato numerico.

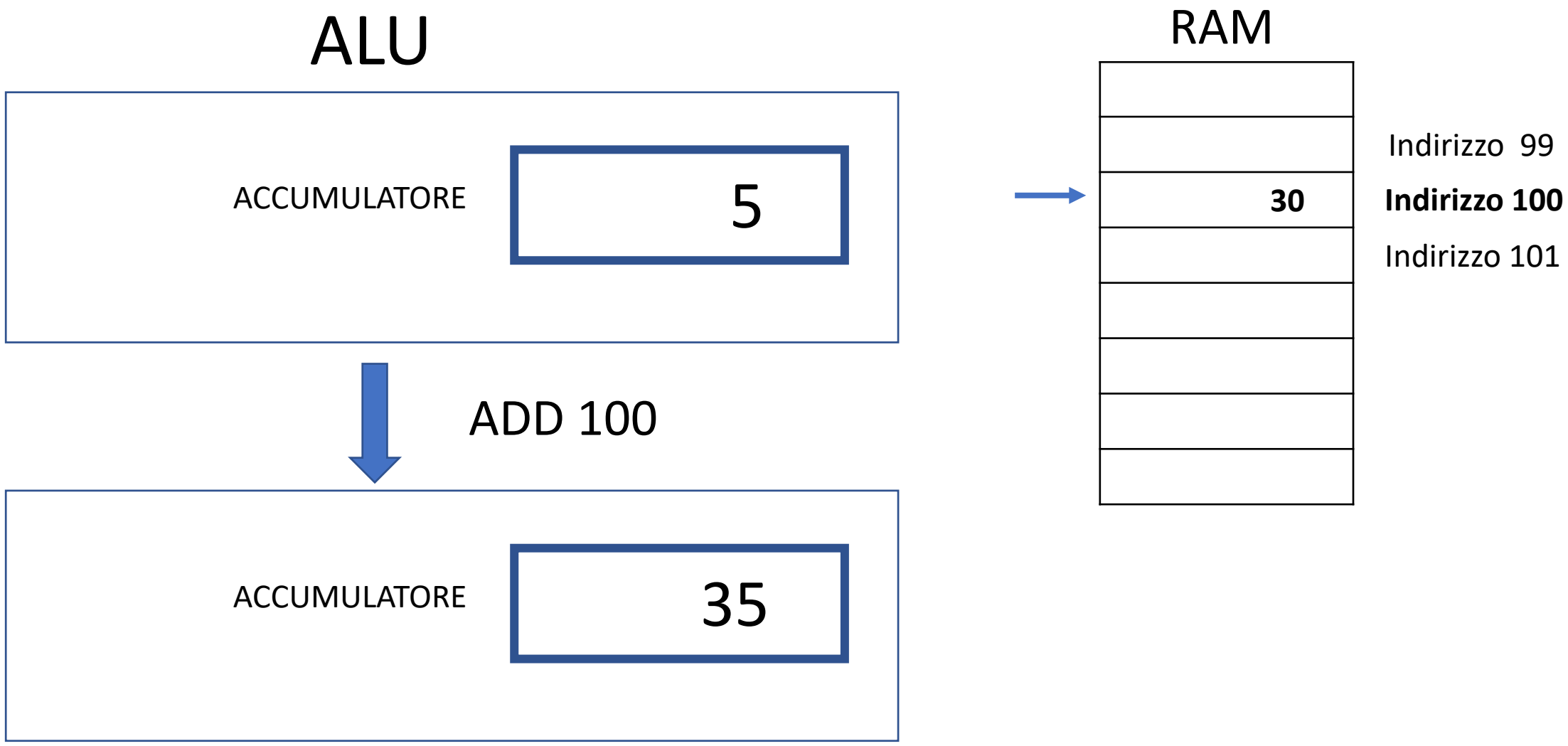
ALU

L’unità ALU comprende normalmente un registro, denominato **accumulatore**, per la memorizzazione temporanea del risultato di operazioni aritmetiche effettuate tra il contenuto numerico dello stesso accumulatore e quello di una specifica locazione di memoria individuata da un indirizzo, in simboli:

$$A \leftarrow A \nabla RAM[ind]$$

Il simbolo ∇ rappresenta una qualsiasi delle operazioni aritmetiche che l'unità ALU è in grado di effettuare.

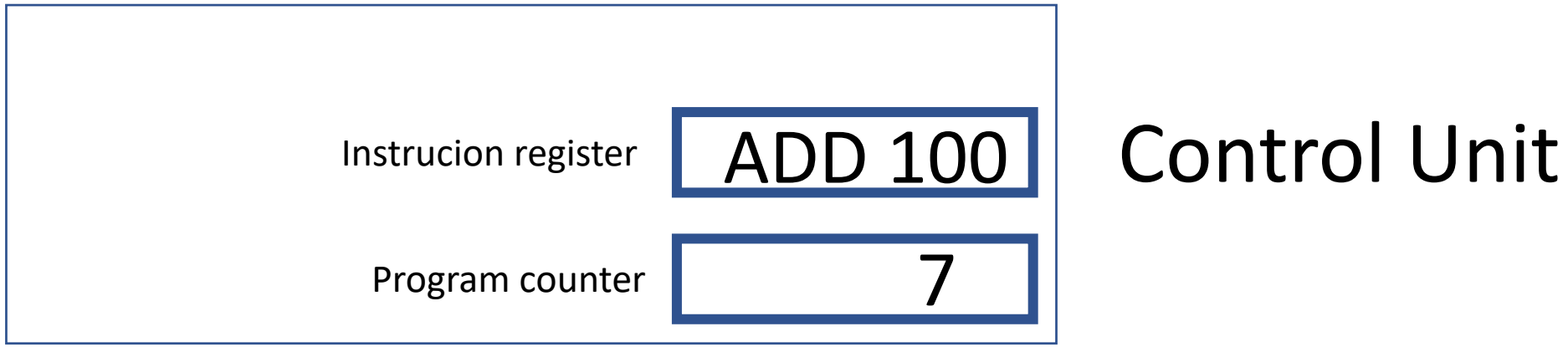
Esempio esecuzione di operazione di ADD:



Completano l'architettura di una *macchina di von Neumann* due registri contenuti nell'unità di controllo:

- il **registro istruzioni** (**IR**, *Instruction Register*), dove un'istruzione prelevata dalla memoria viene interpretata per essere eseguita;
- il **registro contatore di programma** (**PC**, *Program Counter*), il cui valore individua l'indirizzo della locazione di memoria contenente la prossima istruzione da eseguire.

Il valore del registro **PC** viene automaticamente incrementato di una posizione – divenendo l'indirizzo della locazione successiva di memoria – nel corso dell'esecuzione da parte della CPU dell'istruzione stessa.



Il funzionamento di un computer è di conseguenza definito dalla seguente sequenza di fasi di lavoro che viene continuamente ripetuta:

- **FETCH**: l'istruzione contenuta nella locazione di memoria indirizzata dal valore del registro **PC** viene letta e copiata nel registro IR dell'unità di controllo; successivamente viene incrementato il valore del PC in modo da indirizzare l'istruzione successiva da eseguire;
- **EXECUTE**: l'istruzione presente nell'**IR** viene esaminata per riconoscere l'operazione che essa codifica e quindi eseguita.

I termini *fetch* (prelevamento) ed *execute* (esecuzione) identificano le due fasi del ciclo di funzionamento della CPU che un computer attuale è in grado di ripetere anche alcuni miliardi di volte al secondo.

Memorizzando ordinatamente le istruzioni che si intendono eseguire in locazioni successive della memoria RAM e stabilendo il valore iniziale del registro contatore di programma uguale all'indirizzo della prima di esse, si ottiene da parte del computer l'esecuzione automatica della sequenza di operazioni specificata dalle istruzioni. La sequenza di istruzioni da eseguire è denominata **programma**.

Vediamo ora un **linguaggio macchina** eseguito da una computer semplificato a scopo didattico: un solo registro accumulatore, mentre nelle CPU reali ci sono più registri che vengono utilizzati come registri accumulatori e un insieme limitato di OPERAZIONI sia ARITMETICHE che di SALTO

OPERAZIONI di LETTURA e SCRITTURA tra ACCUMULATORE e MEMORIA PRINCIPALE

ISTRUZIONE	DESCRIZIONE ANALITICA	DESCRIZIONE SINTETICA
LOD ind	Carica il valore contenuto nella locazione di memoria di indirizzo **ind** nel registro accumulatore	A <-- RAM[ind]
LOD #num	Carica direttamente il numero specificato nel registro accumulatore	A <-- num
STO ind	Copia il valore contenuto nel registro accumulatore nella locazione di memoria di indirizzo **ind**	RAM[ind] <-- A

OPERAZIONI ESEGUITE DALLA ALU: ADDIZIONE

ISTRUZIONE	DESCRIZIONE ANALITICA	DESCRIZIONE SINTETICA
ADD ind	Addiziona il valore contenuto nel registro accumulatore e il valore contenuto nella locazione di memoria di indirizzo **ind** calcolando la somma nel registro accumulatore	A <-- A + RAM[ind]
ADD #num	Addiziona il valore contenuto nel registro accumulatore e il numero specificato calcolando la somma nel registro accumulatore	A <-- A + num

OPERAZIONI ESEGUITE DALLA ALU: SOTTRAZIONE

ISTRUZIONE	DESCRIZIONE ANALITICA	DESCRIZIONE SINTETICA
SUB ind	Sottrae il valore contenuto nella locazione di memoria di indirizzo **ind** dal valore contenuto nel registro accumulatore calcolando la differenza nel registro accumulatore	A <-- A - RAM[ind]
SUB #num	Sottrae il numero specificato dal valore contenuto nel registro accumulatore calcolando la differenza nel registro accumulatore	A <-- A - num

OPERAZIONI ESEGUITE DALLA ALU: MOLTIPLICAZIONE

ISTRUZIONE	DESCRIZIONE ANALITICA	DESCRIZIONE SINTETICA
MUL ind	Moltiplica il valore contenuto nel registro accumulatore con il valore contenuto nella locazione di memoria di indirizzo **ind** calcolando il prodotto nel registro accumulatore	A <-- A * RAM[ind]
MUL #num	Moltiplica il valore contenuto nel registro accumulatore con il numero specificato calcolando il prodotto nel registro accumulatore	A <-- A * num

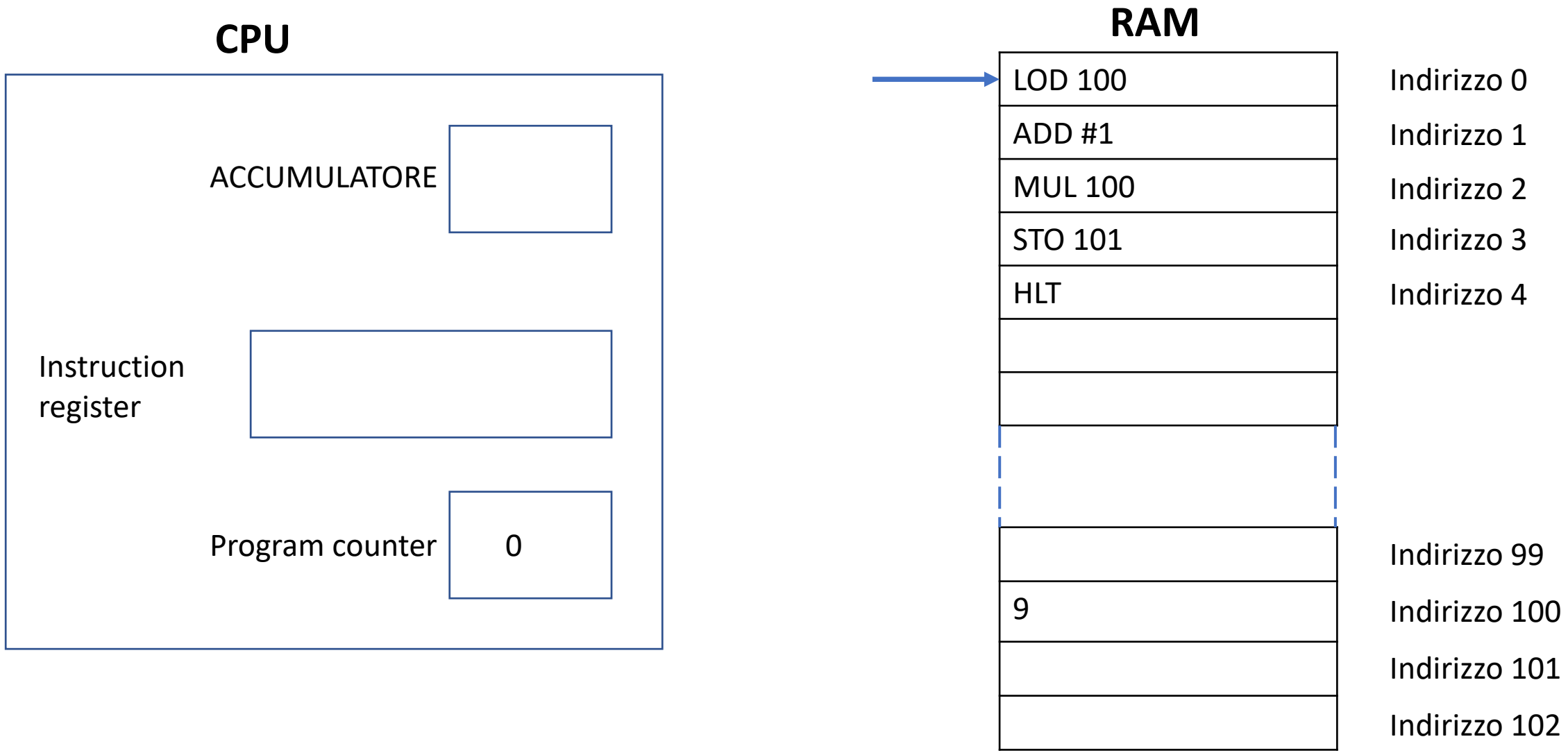
OPERAZIONI ESEGUITE DALLA ALU: DIVISIONE

ISTRUZIONE	DESCRIZIONE ANALITICA	DESCRIZIONE SINTETICA
DIV ind	Divide il valore contenuto nel registro accumulatore per il valore contenuto nella locazione di memoria di indirizzo **ind** calcolando il quoziente nel registro accumulatore (il quoziente è un numero intero anche se il valore contenuto in A non è divisibile per il valore contenuto nella memoria in questo caso viene trascurato il "resto")	A <-- A / RAM[ind]
DIV #num	Divide il valore contenuto nel registro accumulatore per il numero specificato calcolando il quoziente nel registro accumulatore (il quoziente è un numero intero anche se il valore contenuto in A non è divisibile per il numero specificato: in questo caso viene tralasciato il "resto")	A <-- A / num

ISTRUZIONI DI SALTO e ALT

ISTRUZIONE	DESCRIZIONE ANALITICA	DESCRIZIONE SINTETICA
JMP ind	Inserisce il valore **ind** nel registro PC (la prossima istruzione eseguita sarà l'istruzione contenuta nella locazione di memoria avente indirizzo **ind**, anziché l'istruzione contenuta nella posizione di memoria successiva a quella contenente questa istruzione)	PC ← ind
JMZ ind	Inserisce il valore **ind** nel registro PC se il contenuto del registro accumulatore è **zero** (in questo caso la prossima istruzione eseguita sarà l'istruzione contenuta nella locazione di memoria avente indirizzo **ind**, altrimenti sarà l'istruzione contenuta nella locazione di memoria successiva a quella contenente questa istruzione	Se A è zero allora PC ← ind
NOP	Operazione nulla	
HLT	Interrompe il ciclo di funzionamento della macchina	

Il seguente programma – memorizzato nella RAM a partire dall'indirizzo 0 –
effettua il prodotto del numero N inizialmente contenuto nella locazione di indirizzo 100, dato di input,
con il suo successore N + 1 e memorizza il risultato nella locazione di indirizzo 101 (output)



Il programma d'esempio viene eseguito dalla CPU eseguendo l'istruzione all'indirizzo codificato nel *program counter* (PC). L'istruzione viene caricata (*fase di fetch*), incrementato il valore del *program counter*, e poi eseguita (*fase di execute*). Questo ciclo si ripete fino a che non viene letta l'operazione di HLT (alt!). Quindi la CPU è in grado di eseguire l'insieme di istruzioni, scritte in linguaggio macchina, che compongono un programma.

All'indirizzo, https://github.com/checksound/SimulatoreMacchinaVonNeumann, un simulatore di computer in grado di eseguire semplici programmi, utilizzando le semplici istruzioni del linguaggio macchina precedentemente illustrato.

Questo non è diverso da quello che accade quando cliccate (mandate in esecuzione un programma), ad esempio *Microsoft World*: Il **sistema operativo** (Window, Linux , iOS ….) carica in memoria l'eseguibile del programma, prelevandolo dalla memoria secondaria. Da questo punto in poi, il programma viene eseguito dalla CPU secondo lo schema ***fetch-execute***.

Abbiamo ora introdotto un nuovo attore, il S.0., **sistema operativo**, anche lui un programma e quindi eseguito dalla CPU, ma un programma speciale perché permette agli altri programmi istallati di essere caricati in memoria principale prelevandoli dalla memoria secondaria (disco fisso) e quindi di essere eseguiti.

Abbiamo detto che il **sistema operativo** è un programma, anche se speciale, e che quindi anche lui per poter essere eseguito dalla CPU deve essere caricato in memoria principale (RAM). Ma se il sistema operativo si occupa di caricare i programmi da eseguire in memoria, chi si occupa di caricare il sistema operativo in memoria?

Fase di ***bootstrap*** viene chiamata la procedura automatica all'avvio del computer in cui viene caricato in memoria il sistema operativo per poter essere eseguito.

A questo punto il sistema operativo è 'padrone' del computer è può caricare in memoria gli altri programmi su richiesta dell'utente