



Università degli Studi di Cagliari  
Corso di Laurea in Ingegneria Biomedica

# **ELEMENTI DI INFORMATICA**

[https://www.unica.it/unica/page/it/gianluca\\_marcialis](https://www.unica.it/unica/page/it/gianluca_marcialis)

A.A. 2021/2022

Docente: **Gian Luca Marcialis**

**LINGUAGGIO C**  
**Introduzione**

# Sommario

- Linguaggio C: Introduzione
- Tipi di dati in C
- Strutture per articolare gli algoritmi
  - Strutture di controllo
  - Funzioni e procedure
    - Funzioni ricorsive (cenni)
  - Gestione dei file
- I puntatori e strutture dati allocate «dinamicamente»
  - Vettori, pile, code
- Algoritmi notevoli
  - La ricerca
  - L'ordinamento

# Informatica e algoritmi

- “L’informatica è lo studio sistematico degli **algoritmi** che descrivono e trasformano l’informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione”

*Association for Computing Machinery (ACM)*

- **Algoritmo**

- sequenza **precisa** (comprensibile) di **passi elementari** che consentono di realizzare un compito, ovvero risolvere un problema
- passi elementari: eseguibili dall’esecutore dell’algoritmo
- es.: istruzioni di montaggio di un mobile, prelevamento di denaro da un terminale Bancomat, calcolo del massimo comune divisore di due numeri naturali...

# Definizione formale di algoritmo

- Un algoritmo è un insieme **ordinato** di operazioni **non ambigue** ed **effettivamente computabili** che, quando eseguito, **produce un risultato** osservabile e si arresta **in un tempo finito**.
- Le proprietà di un algoritmo sono richiamate dai seguenti termini:
  - “Ordinato”, “non ambigue”: precisione
  - “Effettivamente computabili”, “produce un risultato”: correttezza
  - “In un tempo finito”: efficienza

# Quindi...

- Informatica come **studio degli algoritmi**, dei quali il calcolatore è un ottimo **esecutore**
- Un algoritmo comprensibile al calcolatore è espresso mediante **sequenze di bit**
- Tuttavia è complesso:
  - esprimersi in linguaggio macchina,
  - controllare la macchina della stessa (v. Sistemi Operativi)
- Dall'altra parte il linguaggio naturale non è ideale per scrivere algoritmi rispettandone le proprietà
- Esistono dei **linguaggi** che permettono la scrittura di programmi in una forma accessibile sia a noi che al calcolatore
- Per scrivere un programma esistono **gli ambienti di programmazione**

# Interpreti e compilatori

➤ Per poter eseguire istruzioni di linguaggi di alto livello sulla *macchina fisica sottostante* esistono due possibilità:

- **Interpretazione**, cioè si utilizza un programma chiamato ***interprete*** in grado di eseguire direttamente le istruzioni di alto livello
- **Traduzione**, cioè la sequenza di istruzioni di alto livello viene prima tradotta, utilizzando un programma chiamato *compilatore*, in una corrispondente sequenza di istruzioni della macchina fisica

# Linguaggi interpretati e compilati

- Interpretati

- Basic, Python, Matlab, Lisp, Smalltalk...
- Programma → Istruzione → Traduzione → Esecuzione



- Compilati

- C, C#, C++, Java, Pascal...
- Programma → Traduzione → Istruzione → Esecuzione



# Ambienti di programmazione (IDE)

- Editor
  - serve per scrivere il *programma sorgente*, cioè il **testo** che contiene le istruzioni nel linguaggio prescelto
- Compilatore
  - traduce un programma sorgente in *programma oggetto*, cioè in un programma con un formato molto vicino a quello del linguaggio macchina. Se vi sono errori nella stesura viene avvisato il programmatore e il programma oggetto non viene generato.
- Interprete
  - traduce ed esegue il programma istruzione per istruzione
  - non usa il compilatore, ma un interprete che esegue direttamente il codice sorgente

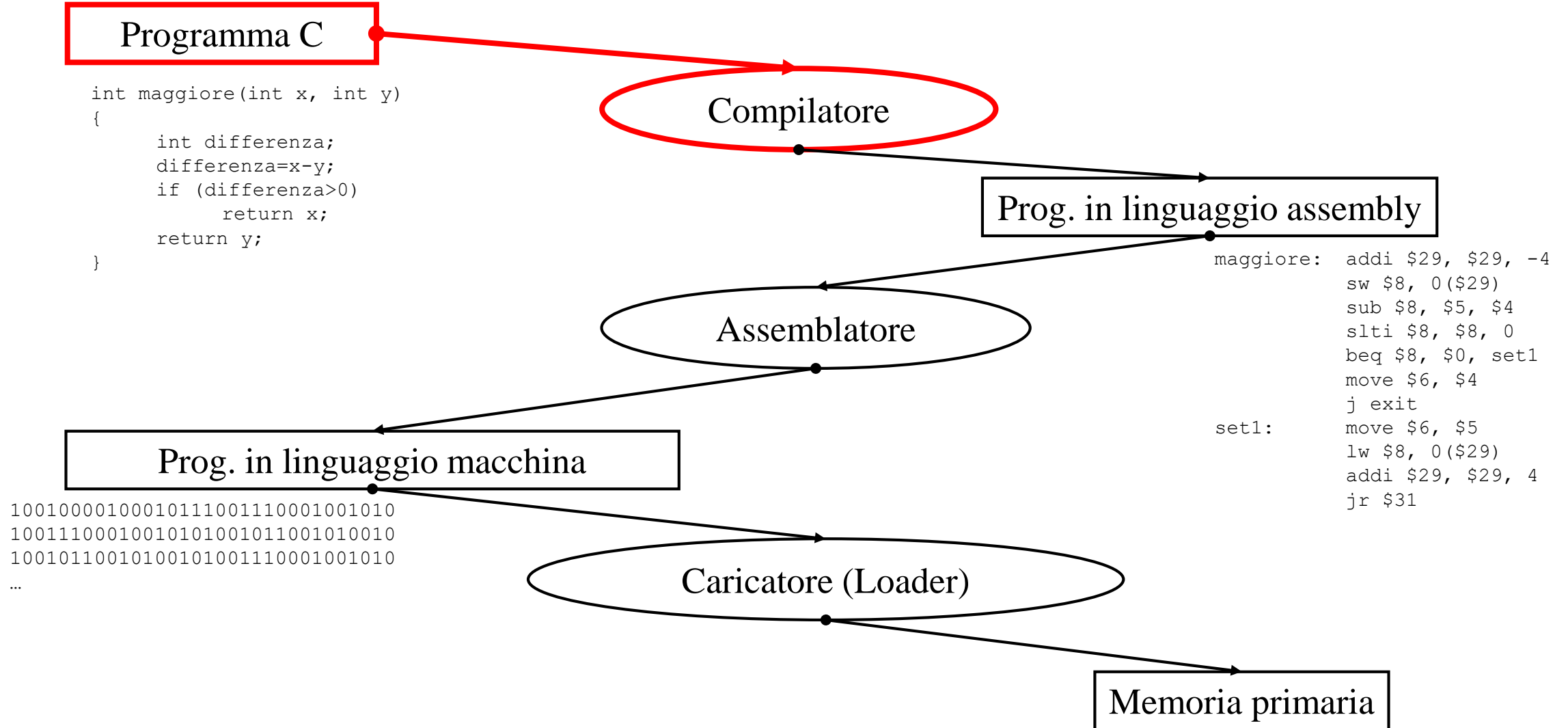
# Ambienti di Programmazione

- Linker
  - collega insieme vari programmi oggetto che fanno parte di un unico programma suddiviso in *moduli* coordinati fra loro, generando il *programma eseguibile*.
- Debugger
  - consente di eseguire il programma passo passo verificando l'esecuzione delle istruzioni del programma sorgente e individuando eventuali errori
- Gli ambienti di programmazione forniscono in genere anche un insieme di *funzioni di libreria*, cioè di algoritmi comuni a molti programmi (es. ordinamento di un vettore di numeri)

# Ricapitoliamo un po'

- Ci siamo soffermati su come l'informatica possa essere lo **studio degli algoritmi**
- Un algoritmo per essere “eseguito” necessita di un modello di calcolo → la macchina di Von Neumann
  - Un algoritmo scritto in un opportuno linguaggio prende il nome di **programma**
- Tuttavia abbiamo rilevato quanto complesso sia esprimersi in linguaggio macchina
  - Nonché effettuare il controllo della stessa (v. Ciclo di esecuzione e Sist. Operativi)
- Dall'altra parte il linguaggio naturale non è ideale per scrivere algoritmi rispettandone le proprietà
- Esistono dei **linguaggi** che permettono la scrittura di programmi in una forma accessibile sia a noi che alla macchina
  - Il **LINGUAGGIO C** è uno di questi

# Programmi: da linguaggio di alto livello a caricamento in memoria



# Lavorare con il C

- GDB Online
  - per scrivere e vedere subito che succede
- DevC++
  - il più semplice per imparare
  - solo Windows ☹️
- Per lavorare e sviluppare (free):
  - CLion
  - Code::Blocks
  - I due ambienti richiedono (e permettono) l'installazione di cygwin, che contiene tutto ciò che serve per far funzionare il sistema

# Esempio: il DevC++

```
1 #include <stdio.h>
2 #include <ctype.h>
3 #include <errno.h>
4 #include <math.h>
5 #include <stdlib.h>
6 #include <string.h>
7 // #include "doag.c"
8 #include "rete.c"
9 // #include "dealloca.c"
10
11 void uso()
12 {
13     printf("\n\n***** GENERATORE DI RETI NEURALI *****");
14     printf("\n\n***** di Gian Luca Marcialis *****");
15     printf("\n\n\nUsa: create param1 ... param7\n");
16     printf("\nparam1 : dimensione dell'INPUT");
17     printf("\nparam2 : dimensione dell'OUTPUT");
18     printf("\nparam3 : numero degli strati della rete");
19     printf("\nparam4.1 ... param4.param3 : numero di neuroni per ogni strato della rete");
20     printf("\nparam5 : minimo valore dei pesi");
21     printf("\nparam6 : massimo valore dei pesi");
22     printf("\nparam7 : nome del file dove memorizzare la rete\n\n");
23     exit(0);
24 }
25
26
27 int main(int argc, char **argv)
28 {
29     int i, stimarg;
30     int numInput, numOutput, numStrati, *dimStrati;
31     float low, high;
32
33     FILE *fp;
34     RETE *rete;
35
36     if(argc==1) uso();
37     printf("\nHai inserito %d argomenti", argc-1);
38
39     ..
```

Compiler (2) Risorse Log di Compilazione Debug Risultati Ricerca Chiudi

Linea	Col...	Unità	Messaggio
		C:\Users\gian\AppData\Local\Temp\cc8aPIV8.o	crea.c:(text+0x2c): undefined reference to 'drand48'
		C:\Dati\Lavoro\Esperimenti\MLP\MLP\collect2.exe	[Error] ld returned 1 exit status

Line: 1 Col: 1 Sel: 0 Lines: 67 Length: 1687 Inserisci Done parsing in 0.14 seconds

# Il nucleo del linguaggio C

- Come tutti i linguaggi, esso è dotato di:
  - Sintassi
  - Semantica
- La sintassi è l'insieme di regole per la costruzione corretta di una “frase”, ovvero di una istruzione
- La semantica è il significato che si dà alla “frase”
- Poiché il linguaggio C deve essere comprensibile al calcolatore, non ci possono essere ambiguità semantiche
  - Una “frase” in C ha sempre lo stesso significato

# Elementi base del C

- Identificatori simbolici
  - Pubblici
    - Rappresentano aree di memoria assegnate a certi dati dal programmatore
      - Es. un certo valore
    - ...oppure a certi gruppi di istruzioni che realizzano un determinato compito (funzioni)
  - Privati (non utilizzabili dal programmatore)
    - Parole-chiave
      - Istruzioni base del linguaggio
    - Nomi di “funzioni” di libreria
      - Gruppi di istruzioni che realizzano un compito accessibili tramite interfacce
        - Es. istruzioni aritmetiche complesse, stampa a schermo
    - Direttive
      - Es. attivazione di un certo insieme di funzioni di libreria

# Il mio primo programma C

- Si scriva in linguaggio C un programma che stampi su Standard Output (es. video) il seguente messaggio:

`Ciao, mondo!`

# Il mio primo programma in C

```
/*Il mio primo programma in C*/  
  
#include <stdio.h>  
  
int main()  
{  
    printf("Ciao, mondo!");  
    return 0;  
}
```

# Il mio primo programma in C

```
/*Il mio primo programma in C*/
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
printf("\nCiao, mondo!\n");
```

```
return 0;
```

```
}
```

Questo è un

**commento:** non

← contiene istruzioni dell'algoritmo ma in genere serve per spiegare cosa fa una certa parte di programma (codice)

Un **commento** viene inserito fra i simboli:

/ \* → inizio commento

\* / → fine commento

# Il mio primo programma in C

```
/*Il mio primo programma in C*/
```

```
#include <stdio.h>
```



```
int main()
```

```
{
```

```
    printf("\nCiao, mondo!\n");
```

```
    return 0;
```

```
}
```

E' una **direttiva**:  
serve in questo  
caso ad attivare  
un insieme di  
funzioni già  
pronte per  
comunicare con i  
periferici di nome  
"stdio.h".

Una direttiva è un  
identificatore  
simbolico  
preceduto dal  
carattere **#**.

# Il mio primo programma in C

```
/*Il mio primo programma in C*/
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
printf("Ciao, mondo!");
```

```
return 0;
```

```
}
```

E' l' "intestazione"  
del programma.

Il termine privato  
"main" indica  
appunto che la  
sequenza di  
istruzioni compresa  
entro le parentesi  
graffe è l'algoritmo

L'identificatore **int** è una **parola chiave**: al  
termine del programma è attesa un'istruzione  
che segnali la fine del programma (divenuto  
processo) attraverso un valore numerico

# Il mio primo programma in C

```
/*Il mio primo programma in C*/  
  
#include <stdio.h>  
  
int main()  
{  
    printf("Ciao, mondo!");  
    return 0;  
}
```

E' un'istruzione composta da una **funzione** presente nella libreria "stdio.h".

L'identificatore **printf** è il nome della funzione, e prevede che si stampi a video la sequenza di caratteri fra virgolette "Ciao, mondo!".

# Il mio primo programma in C

```
/*Il mio primo programma in C*/
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Ciao, mondo!");
```

```
    return 0;
```

```
}
```

E' un'altra **parola-chiave** del linguaggio.

Significa che l'algoritmo ha termine restituendo un valore intero atteso (in questo caso, **0**).

# Riassumendo

- Un programma C può essere caratterizzato da una o più righe di **commento** (*/\* ... \*/*)
- Seguono opportune **direttive** (*#include*) per l'attivazione di **funzioni di libreria** (*stdio.h*) che possono essere utili per la scrittura del programma
- L'algoritmo viene inserito **tra due parentesi graffe {}** precedute dall'**intestazione di funzione principale** (*int main()*)
- La scrittura dell'algoritmo può richiedere l'uso delle **funzioni** presenti nella libreria dichiarata (*printf()*)
- L'algoritmo termina con un'istruzione di “ritorno” o “fine” (*return*)

# **L'istruzione di assegnamento**

- Sia dato il seguente problema:

**Scrivere un programma in C che riceva da tastiera due valori interi e stampi su video la loro somma**

# Soluzione

```
/*Programma per la stampa a video della somma di due numeri*/

#include <stdio.h>
int main()
{
    int a, b, somma; /* dichiarazione di variabile */
    printf("Inserire due valori interi da sommare\n");
    scanf("%d%d", &a, &b); /*input formattato*/ /*indirizzo di...*/
    somma = a + b; /*assegnamento*/
    printf("%d + %d = %d\n", a, b, somma); /*output formattato*/
    return 0;
}
```

# La soluzione presentata

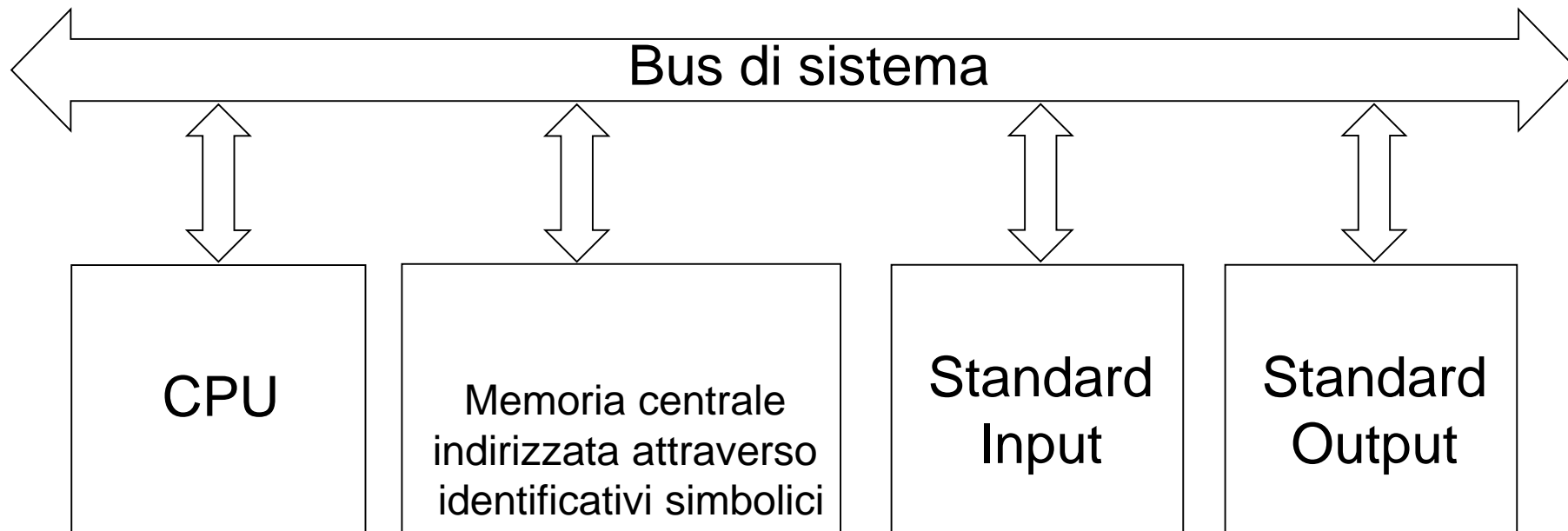
- Cosa cambia rispetto al primo programma in C?
- La sequenza di istruzioni all'interno delle graffe (è un altro algoritmo)
- In particolare, viene chiamata la funzione `scanf` che serve a leggere da *standard input* (la tastiera) dei valori
  - Tali valori sono identificati dalla coppia simboli `%d` che identificano ciascuno un valore intero
  - I valori letti da tastiera vengono **assegnati** alle **variabili** `a`, `b`
- Infine la somma di `a` con `b` viene **assegnata** ad un'altra variabile `somma`...
- ... che viene stampata su *standard output* (il monitor) tramite la funzione `printf`

# Dichiarazione di variabili

- E' detta **parte dichiarativa** di un programma C il punto in cui si elencano tutti gli **identificatori** di un certo **tipo** da utilizzare
  - Solitamente precedente il corpo del programma (l'algoritmo)
- Questi **identificatori** prendono il nome di **variabili**
- Esempio: `int a, b, somma;`
  - Stiamo dichiarando tre variabili di tipo "intero" con nome a, b, somma
- Il C mette a disposizione del programmatore diversi tipi di dato, ben più potenti del semplice `int`

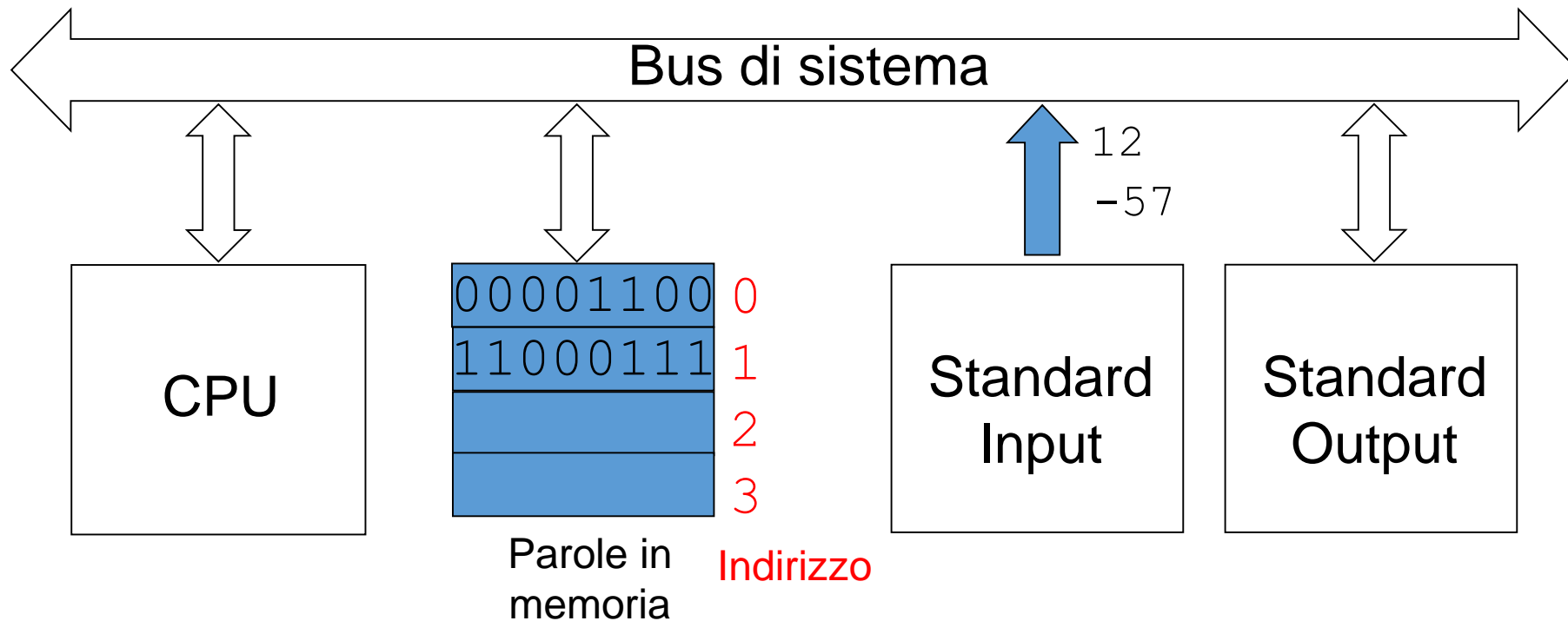
# Una “macchina virtuale” per il C

- Per l'esecuzione di un programma C, assumeremo che la nostra macchina di Von Neumann (il nostro PC) sia così organizzata:



# Cos'è una variabile

- Fisicamente (lato calcolatore), è una o più parole di memoria che devono contenere un valore associato ad una data rappresentazione binaria



# Cos'è una variabile

- Logicamente (lato programmatore), è un identificatore simbolico finalizzato a contenere valori assegnati da programma o da input esterno, elaborati durante l'esecuzione del programma, trasferiti in output

```
/*Variabile assegnata da input esterno*/  
scanf ("%d" , &x) ;
```

```
/*Variabile assegnata da programma*/  
y=10;
```

```
/*Variabile elaborata durante l'esecuzione*/  
x=x*y;
```

```
/*Variabile trasferita in output*/  
printf ("%d" , x) ;
```

# Cos'è una variabile

- Lato programmatore + lato fisico
  - E' un contenitore di valori di tipo definito (es. interi)
  - E' una locazione di memoria assegnata ed indirizzata mediante il suo nome
- L'indirizzo di memoria specifico si ottiene anteponendo l'operatore & al nome della variabile
  - Es. &a == indirizzo di a (mentre a indica il suo generico contenuto)
- Una variabile necessita di essere **dichiarata** perché essa trovi posto in memoria centrale, ovvero sia **allocata**
  - `int a, b, somma;`
- Ad essa può quindi essere **assegnato** un valore con apposita istruzione
  - `somma = a + b;`
  - `somma = 100;`

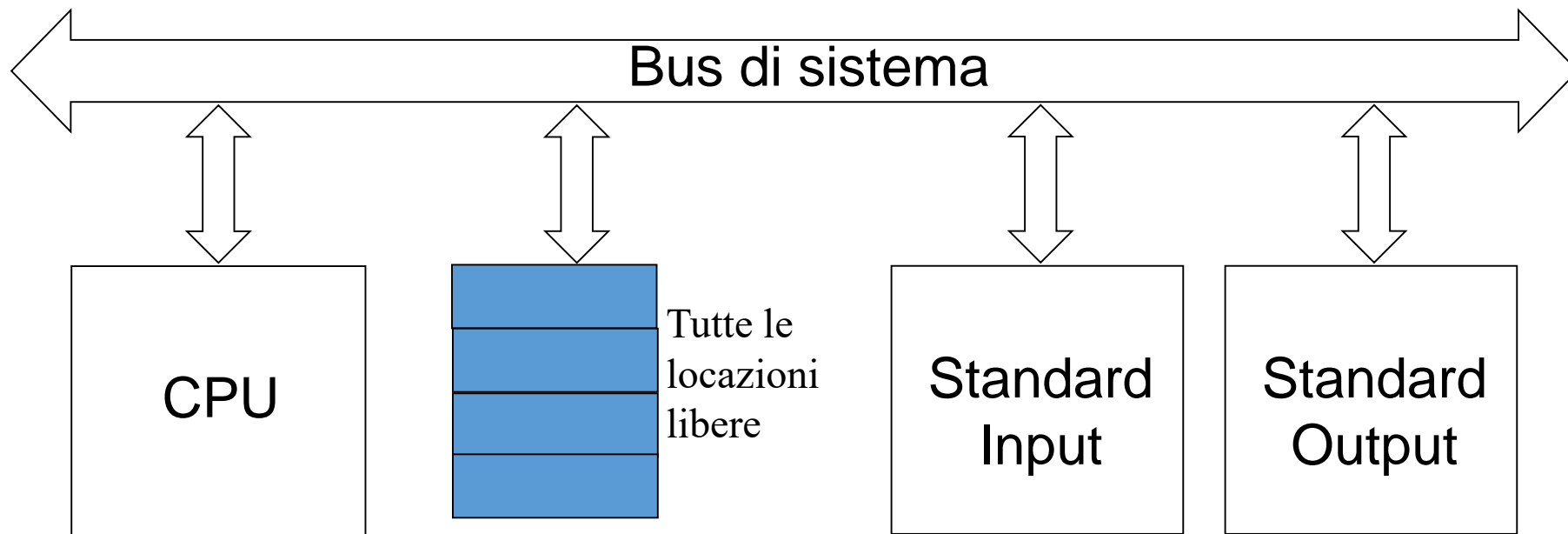
# Il programma della somma

```
/*Programma per la stampa a video della somma di due numeri*/

#include <stdio.h>
int main()
{
    int a, b, somma;
    printf("Inserire due valori interi da sommare\n");
    scanf("%d%d",&a,&b);
    somma = a + b; /*assegnamento*/
    printf("%d + %d = %d\n",a,b,somma);
    return 0;
}
```

# Esecuzione del programma

- Lo stato della “macchina virtuale C” prima della dichiarazione di a, b, somma



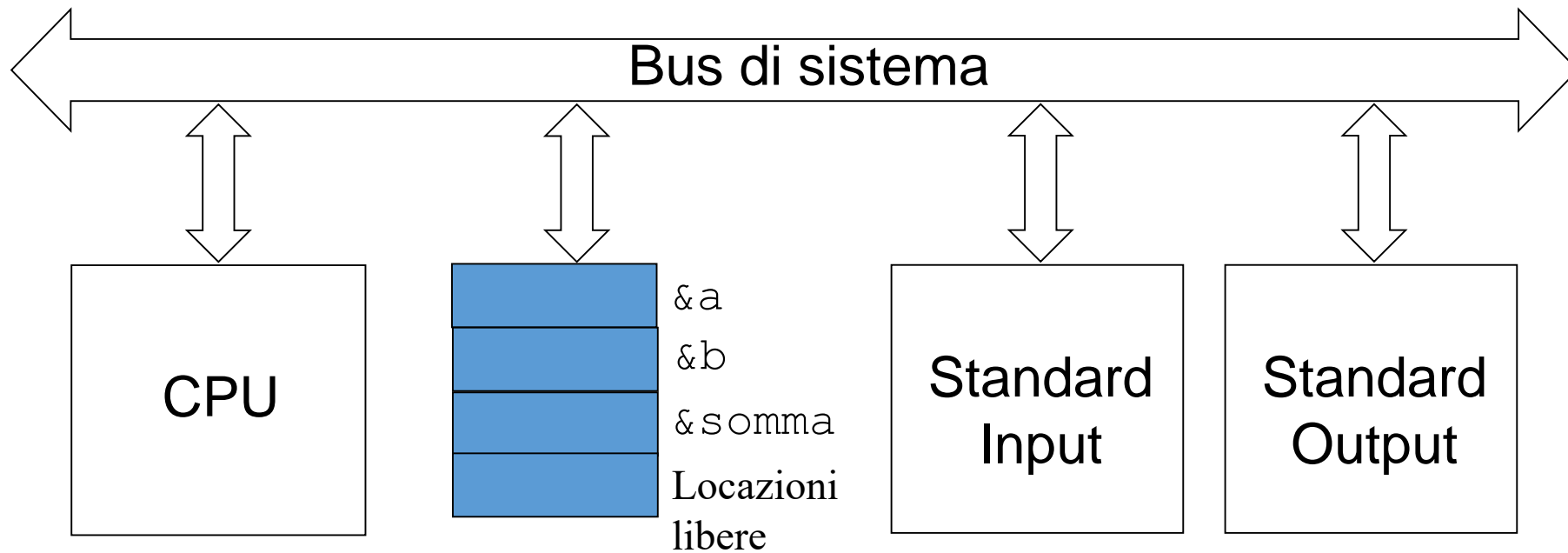
# Dichiarazione di a, b, somma

```
/*Programma per la stampa a video della somma di due numeri*/

#include <stdio.h>
int main()
{
    int a, b, somma;
    printf("Inserire due valori interi da sommare\n");
    scanf("%d%d",&a,&b);
    somma = a + b; /*assegnamento*/
    printf("%d + %d = %d\n",a,b,somma);
    return 0;
}
```

# Esecuzione della dichiarazione

- Lo stato della “macchina virtuale C” dopo la dichiarazione di a, b, somma



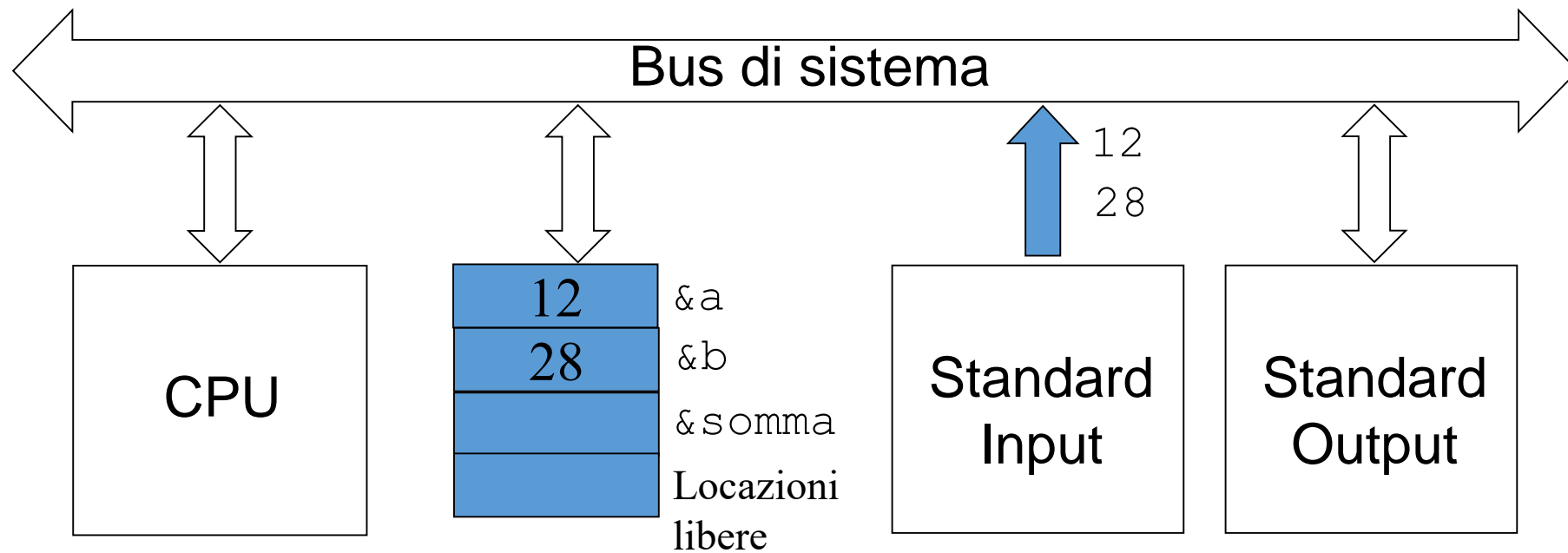
# Lettura di a, b

```
/*Programma per la stampa a video della somma di due numeri*/

#include <stdio.h>
int main()
{
    int a, b, somma;
    printf("Inserire due valori interi da sommare\n");
    scanf ("%d%d", &a, &b) ;
    somma = a + b; /*assegnamento*/
    printf("%d + %d = %d\n", a, b, somma) ;
    return 0;
}
```

# Esecuzione della lettura

- Lo stato della “macchina virtuale C” dopo la lettura di due valori numerici dalla tastiera



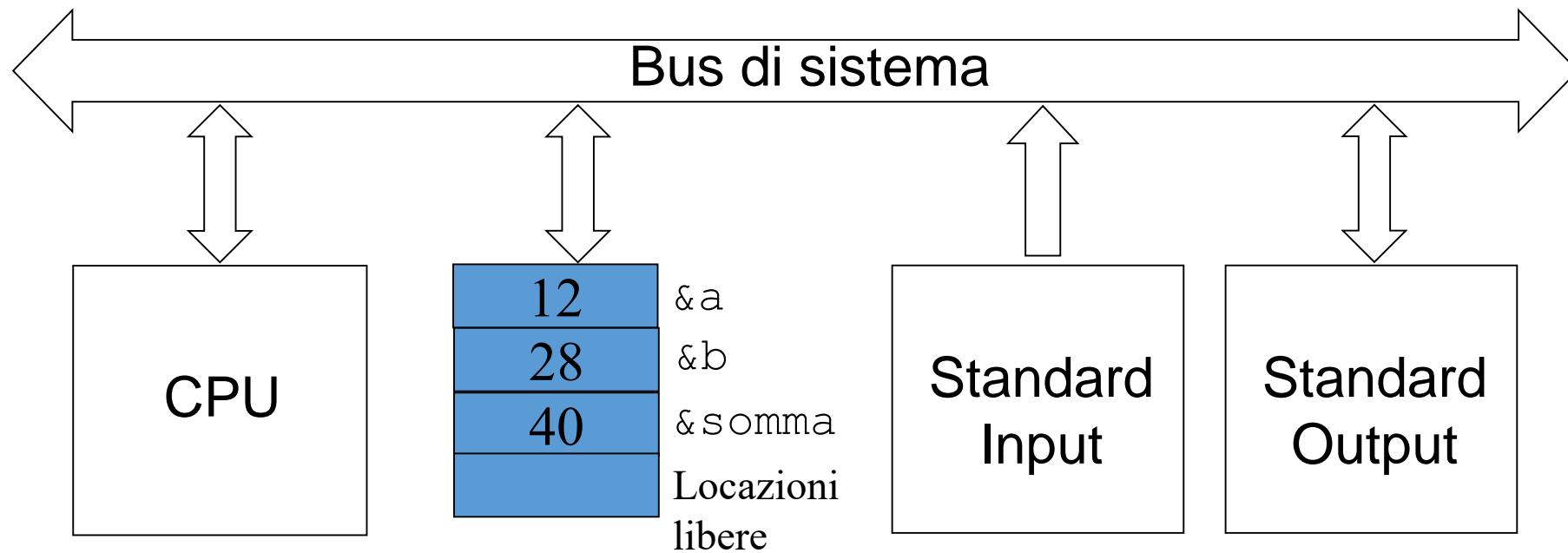
# Assegnamento di somma

```
/*Programma per la stampa a video della somma di due numeri*/

#include <stdio.h>
int main()
{
    int a, b, somma;
    printf("Inserire due valori interi da sommare\n");
    scanf("%d%d", &a, &b);
    somma = a + b; /*assegnamento*/
    printf("%d + %d = %d\n", a, b, somma);
    return 0;
}
```

# Esecuzione dell'assegnamento

- Lo stato della “macchina virtuale C” dopo l'assegnamento della variabile `somma`



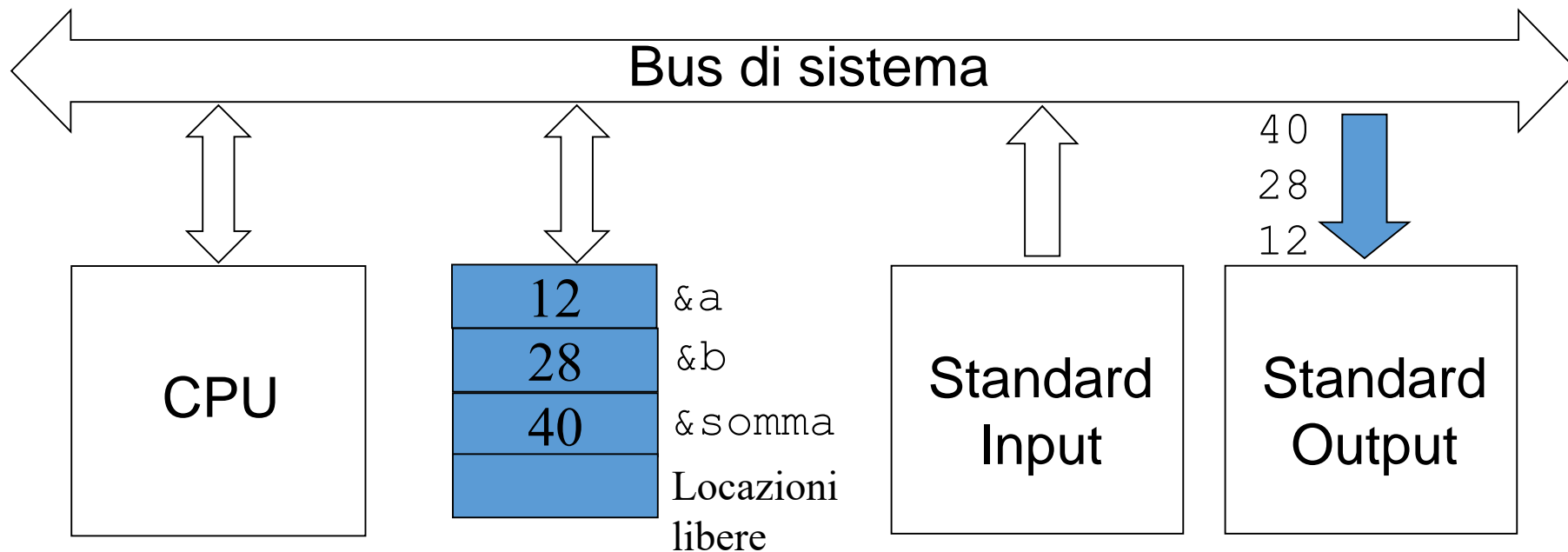
# Stampa di a, b, somma

```
/*Programma per la stampa a video della somma di due numeri*/

#include <stdio.h>
int main()
{
    int a, b, somma;
    printf("Inserire due valori interi da sommare\n");
    scanf("%d%d",&a,&b);
    somma = a + b; /*assegnamento*/
    printf("%d + %d = %d\n",a,b,somma);
    return 0;
}
```

# Esecuzione della stampa

- Lo stato della “macchina virtuale C” dopo la stampa di somma a video



# Struttura generale di un programma C

- Commenti
- Direttive (anche dichiarazione di costanti)
- `int main()`
- {
  - PARTE DI DICHIARAZIONE DELLE VARIABILI;
  - ALGORITMO VERO E PROPRIO;
  - `return 0;`
- }

# Operazioni aritmetiche e booleane elementari su variabili intere

- Somma, sottrazione, prodotto, quoziente, resto:  $+$   $-$   $*$   $/$   $\%$
- «Uguale a» e «diverso da»:  $==$   $!=$
- Esempi:  

```
area_quadrato = lato*lato;  
uguale = a==b; /* uguale vale 1 se il valore di a è uguale a quello di b */  
diverso = a!=b; /* diverso vale 1 se il valore di a è diverso da quello di b */
```

# Esercizi

- Scrivere un programma C che riceva da tastiera un valore intero del lato di un quadrato e ne stampi a video l'area ed il perimetro
- Scrivere un programma C tale che, assegnati da tastiera i parametri di una retta  $a$  e  $b$ , stampi su video l'espressione  $y = ax + b$  dove al posto di  $a$  e  $b$  figurino i valori assegnati. Assegnando poi, sempre da tastiera, un valore intero specifico per  $x$ , stampi il corrispondente valore di  $y$ .
  - Es. se da tastiera immetto 2 e 3, il programma stampa  $y = 2x + 3$

# Per saperne di più

- Ceri, Mandriola, Sbattella, *Informatica – arte e mestiere*, Capp.3-4, McGraw-Hill
- Kernighan, Ritchie, *Il linguaggio C*, Cap. 1, Pearson-Prentice Hall