Linguaggi di Basso Livello

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Motivazione e Sommario

- Nonostante il C sia un linguaggio ad alto livello, risente di alcune limitazioni che sono intrinseche nell'architettura del calcolatore
- Rivisitiamo un linguaggio a basso livello "semplificato" per evidenziarle

Sommario:

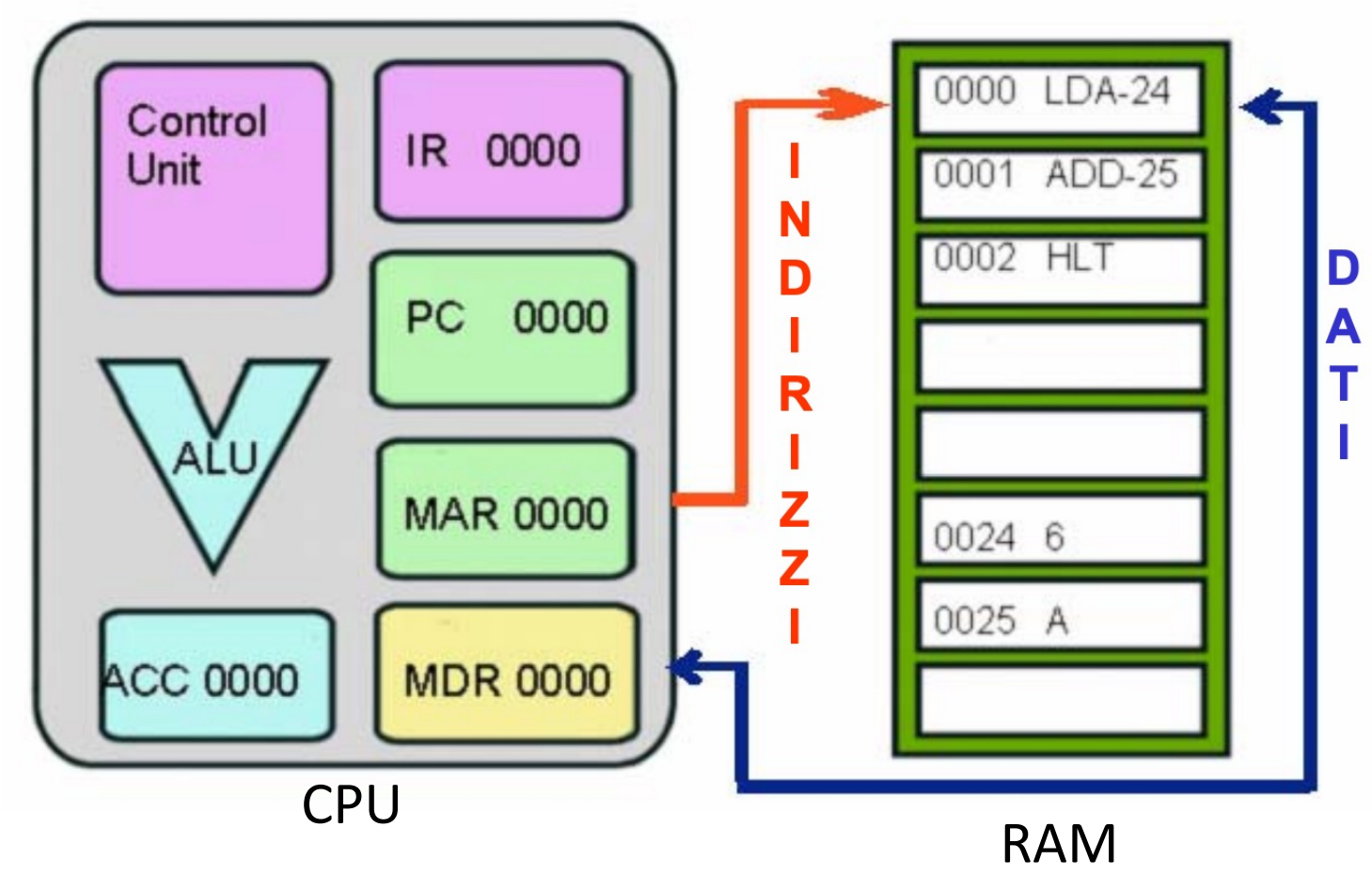
- Rappresentazione di numeri al calcolatore
  - interi con segno in complemento a 2
  - reali
- Linguaggio macchina
  - ciclo FDE
  - istruzioni di base disponibili

Architettura dell'elaboratore

Università degli Studi di Padova

Il Calcolatore è composto da:

- CPU (central processing unit), "il cervello" che esegue i calcoli
- RAM (memoria con i dati e le istruzioni dei nostri programmi in esecuzione)
- Memoria secondaria (dove vengono salvati permanentemente, su file, i programmi quando non sono in esecuzione)
- Periferiche di input/output



Rappresentazione di Numeri nel Calcolatore

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Tipi di Dato

- La memoria del calcolatore è formata da sequenze di celle, i bit, che possono assumere i valori {0,1}

- Il calcolatore è in grado di elaborare differenti tipi di informazione:
  - numeri, caratteri, immagini, suoni, video

Informazione = Dati + Interpretazione

- Poiché ogni tipo di informazione è rappresentato da una sequenza di bit in memoria, dobbiamo sempre sapere la corretta codifica per leggere/scrivere diversi tipi di dato

Rappresentazione di Interi Positivi

- Una sequenza di cifre in base 10 forma un numero secondo la seguente convenzione: $374 = 3 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0$

- Per determinare il valore di un numero binario positivo, si utilizza lo stesso algoritmo dove però la base è 2:

$$(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = 13$$

Rappresentazione di Interi Positivi

- Il numero di configurazioni diverse di n bit è 2^n, per cui si riescono a rappresentare al massimo 2^n numeri diversi.

- Il numero più grande rappresentabile con n bit è $2^n - 1$ (perché si conta anche lo 0). Il calcolatore non può rappresentare infiniti numeri!

- Se il risultato di un'espressione dà un numero oltre il valore massimo rappresentabile, si ottiene un errore di overflow

Rappresentazione di Interi Positivi

- Trasformazione da base 10 a base k

1. Dividere il numero per k
2. tenete traccia del resto
3. se il quoziente è maggiore di 0 ripetere il passo 1 con il quoziente
4. scrivere i resti nell'ordine inverso rispetto al quale sono stati ottenuti

numero	quoziente	resto
43/2	21	1
21/2	10	1
10/2	5	0
5/2	2	1
2/2	1	0
1/2	0	1

$$(43)_{10} = (101011)_2$$

Rappresentazione di Interi con Segno

- Il bit più a sinistra rappresenta il segno: 0 = positivo, 1 = negativo
- se si utilizzano n bit, si riescono a rappresentare tutti i numeri

$$x. -(2^{n-1}-1) \leq x \leq 2^{n-1}-1 \Leftrightarrow -2^{n-1}+1 \leq x \leq 2^{n-1}-1$$

Es. con 4 bit si rappresenta [-7,7]

Binario	Decimale		Binario	Decimale
0000	0		1000	0
0001	1		1001	-1
0010	2		1010	-2
0011	3		1011	-3
0100	4		1100	-4
0101	5		1101	-5
0110	6		1110	-6
0111	7		1111	-7

Rappresentazione in Complemento a 2

- I numeri positivi sono rappresentati in modo “standard” (con gli algoritmi che abbiamo appena visto), utilizzando n bit

- I numeri negativi sono rappresentati “in complemento a 2”, ovvero si somma 2^n al numero e poi si rappresenta in modo “standard”. Es. n = 4

$$(-3)_{10} \rightarrow 2^4 - 3 = 16 - 3 = (13)_{10} = (1101)$$

Rappresentazione in Complemento a 2

Rappresentazione in complemento a 2 con 4 bit

Binario	Decimale		Binario	Decimale
0000	0		1000	-8
0001	1		1001	-7
0010	2		1010	-6
0011	3		1011	-5
0100	4		1100	-4
0101	5		1101	-3
0110	6		1110	-2
0111	7		1111	-1

il primo bit (a sinistra) indica il segno

Una sola rappresentazione per lo zero

Rappresentazione di Interi con Segno

- I numeri interi positivi sono rappresentati all'interno dell'elaboratore utilizzando un multiplo del byte=8 bit (generalmente 4 o 8 byte)
  - dipende dalla singola architettura
- L'istruzione sizeof(int) restituisce il numero di byte (celle di memoria) occupati da un int (intero)
- Il file limits.h (#include <limits.h>) riporta una serie di costanti numeriche
  - INT_MAX: il massimo intero (positivo) rappresentabile sul calcolatore
  - INT_MIN: il minimo intero rappresentabile (negativo)

Rappresentazione di Interi con Segno

- Il calcolatore usa la rappresentazione in complemento a 2
- INT_MAX = 2147483647 (sul mio calcolatore)
- INT_MAX + 1 = ?

Quiz

Rappresentazione di Interi con Segno

- INT_MAX = 2147483647 (sul mio calcolatore)

- INT_MAX + 1 = -2147483648
- Il calcolatore usa la rappresentazione in complemento a 2

- Il C non fornisce un meccanismo automatico per controllare se il risultato di un'addizione sia maggiore di INT_MAX
  - è compito del programmatore

Definizione di Macro

- Cos'è e come si definisce INT_MAX?
- INT_MAX è una macro:

#define INT_MAX 2147483647

- Il preprocessore sostituirà ogni occorrenza di INT_MAX nel codice con il valore 2147483647

```c
#include <stdio.h>

#define VOTO_MAX 30

int main () {

  printf("Il voto massimo che si può prendere");
  printf(" ad un esame è %d\n", VOTO_MAX);

}
```

cosa vede il compilatore

```c

int main () {

  printf("Il voto massimo che si può prendere");
  printf(" ad un esame è %d\n", 30);

}
```

Definizione di Macro

- Sintassi rigida: #define<SPAZIO><IDENTIFICATORE><SPAZIO><VALORE>
- il carattere # deve iniziare nella prima colonna, seguito da define (senza spazi in mezzo), poi uno spazio, poi l'identificatore, ancora uno spazio ed infine il valore
- La macro non è seguita da un ; perché non è un comando C, ma una direttiva per il preprocessore

```c
#include <stdio.h>

#define VOTO_MAX 30

int main () {

  printf("Il voto massimo che si può prendere");
  printf(" ad un esame è %d\n", VOTO_MAX);

}
```

cosa vede il compilatore

```c

int main () {

  printf("Il voto massimo che si può prendere");
  printf(" ad un esame è %d\n", 30);

}
```

Macro: Discussione

- Poter scrivere INT_MAX al posto di 2147483647 ha vari vantaggi:

👍
- se cambiamo architettura dobbiamo solamente modificare la riga #define, non tutte le occorrenze di 2147483647 in tutti i nostri programmi! Poiché INT_MAX è definita in limits.h non dobbiamo fare niente!
  - #include <limits.h>

👍
- Un nome è solitamente più informativo di un numero (INT_MAX mi suggerisce che sia il più grande intero, 2147483647 potrebbe non dirmi niente)
  - Scriviamo programmi spesso in squadra, per cui se un collega (o noi stessi qualche mese dopo) deve leggere il nostro codice, non deve perdere tempo inutilmente!

👎
- Le macro possono rendere un errore nel nostro codice più difficile da identificare
  - vedremo un esempio quando introdurremo le espressioni

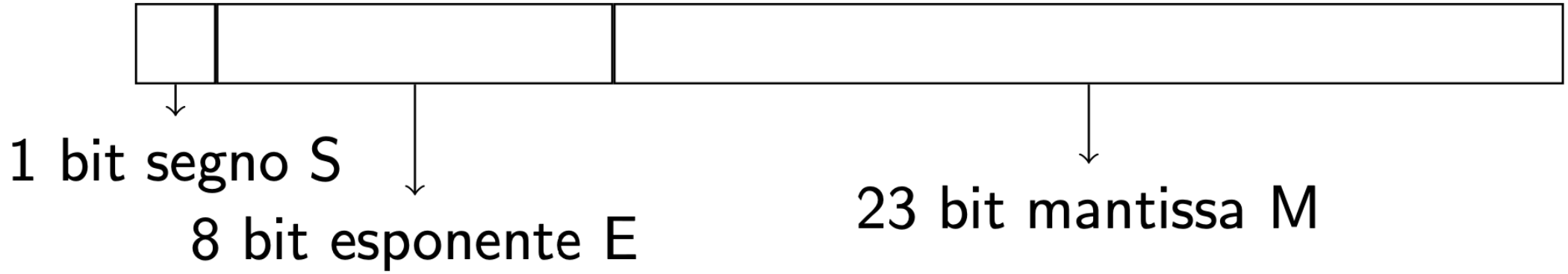
Rappresentazione di Interi con Segno

- Varianti del tipo int

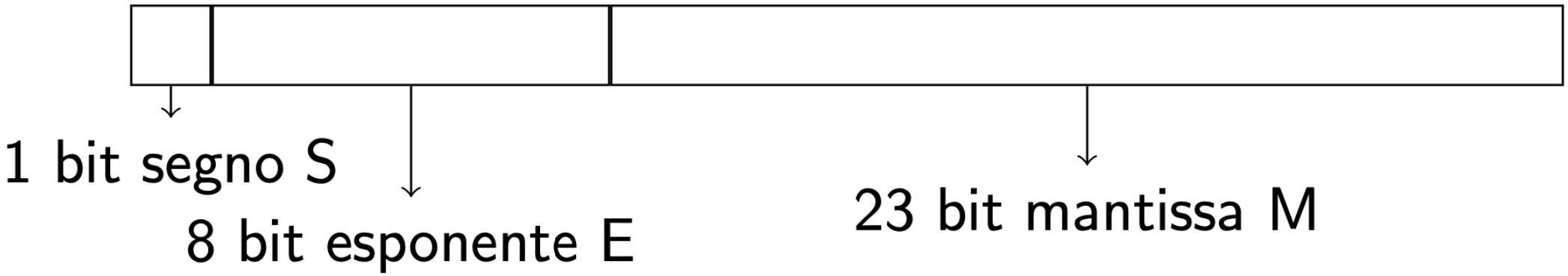
Nome	Descrizione	Byte	Valore Min	Valore Max	formato in printf
int	intero	4	INT_MIN	INT_MAX	printf(“%d”, x)
long	intero che usa il doppio dei byte	8	LONG_MIN	LONG_MAX	printf(“%ld”, x)
short	intero che usa la metà dei byte	2	SHRT_MIN	SHRT_MAX	printf(“%hd”, x)
unsigned int	un intero positivo	4	0	UINT_MAX	printf(“%u”, x)
unsigned long	un long positivo	8	0	ULONG_MAX	printf(“%lu”, x)

Numeri Reali

- I numeri reali utilizzano la rappresentazione in virgola mobile
- Si basa sulla notazione scientifica $1.40 \cdot 10^2 = 140$ (notate che c'è solo una cifra intera, ovvero la notazione è normalizzata)
- Lo standard IEEE 754 prevede vari tipi di numeri in virgola mobile, tra cui:
  - singola precisione (32 bit) e doppia precisione (64 bit)
- i numeri a singola precisione hanno il seguente formato:



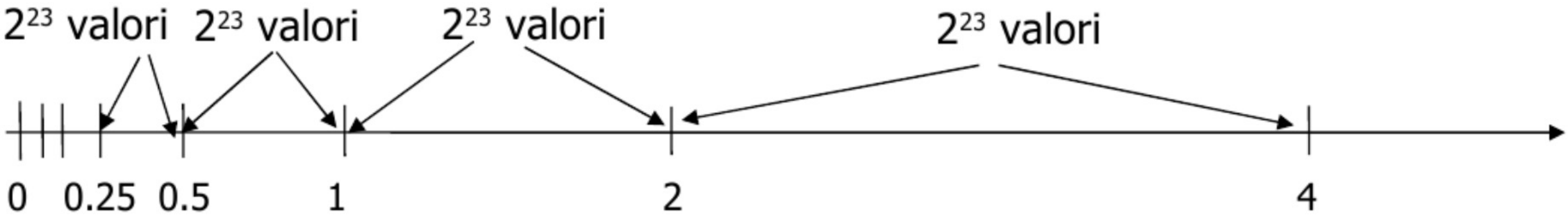
Numeri Reali



- Il formato IEEE 754 è $(-1)^S \cdot 1.M \cdot 2^{E-127}$
- 1.M, la mantissa indica il numero vero e proprio in forma normalizzata
- 2^{E-127} indica dove mettere la virgola (moltiplicare/dividere per 2 un numero binario significare spostare a destra/sinistra la virgola di una posizione)

Numeri Reali

- Intervallo rappresentabile in singola precisione: circa da $1.4 \cdot 10^{-45}$ a $6.81 \cdot 10^{38}$
- in totale si riescono a rappresentare 2^{32} numeri distinti (metà positivi, metà negativi)



- I numeri rappresentabili non sono distribuiti uniformemente,ma sono più densi vicino allo zero.

Numeri Reali in C

- Tipo reale in singola precisione: float x [printf("%f",x)]
- Tipo reale in doppia precisione: double x [printf("%f",x)]
- Es. double x = 3.2;
- Esiste anche il tipo long double x [printf("%Lf",x)]
- Per controllare il numero di cifre decimali stampate:
  - printf("%.2f", 3.1415); → 3.14. Ma printf("%.2f", 3.1475); → 3.15 (nella stampa il numero viene arrotondato
- Attenzione: poiché i reali non hanno precisione infinita, può darsi che, confrontando due espressioni reali (complesse) equivalenti, l'operatore di uguaglianza restituisca falso per colpa delle approssimazioni durante calcoli intermedi.

Numeri Reali in C


```
1  #include <stdio.h>
2
3  int main () {
4
5    printf("%.16f + %.16f = %.16f\n", 3.1415, 0.0000000000000001,
6                                       3.1415+0.0000000000000001);
7
8  }
```

```
$ ./a.out
3.1415000000000002 + 0.0000000000000001 = 3.1415000000000002
```

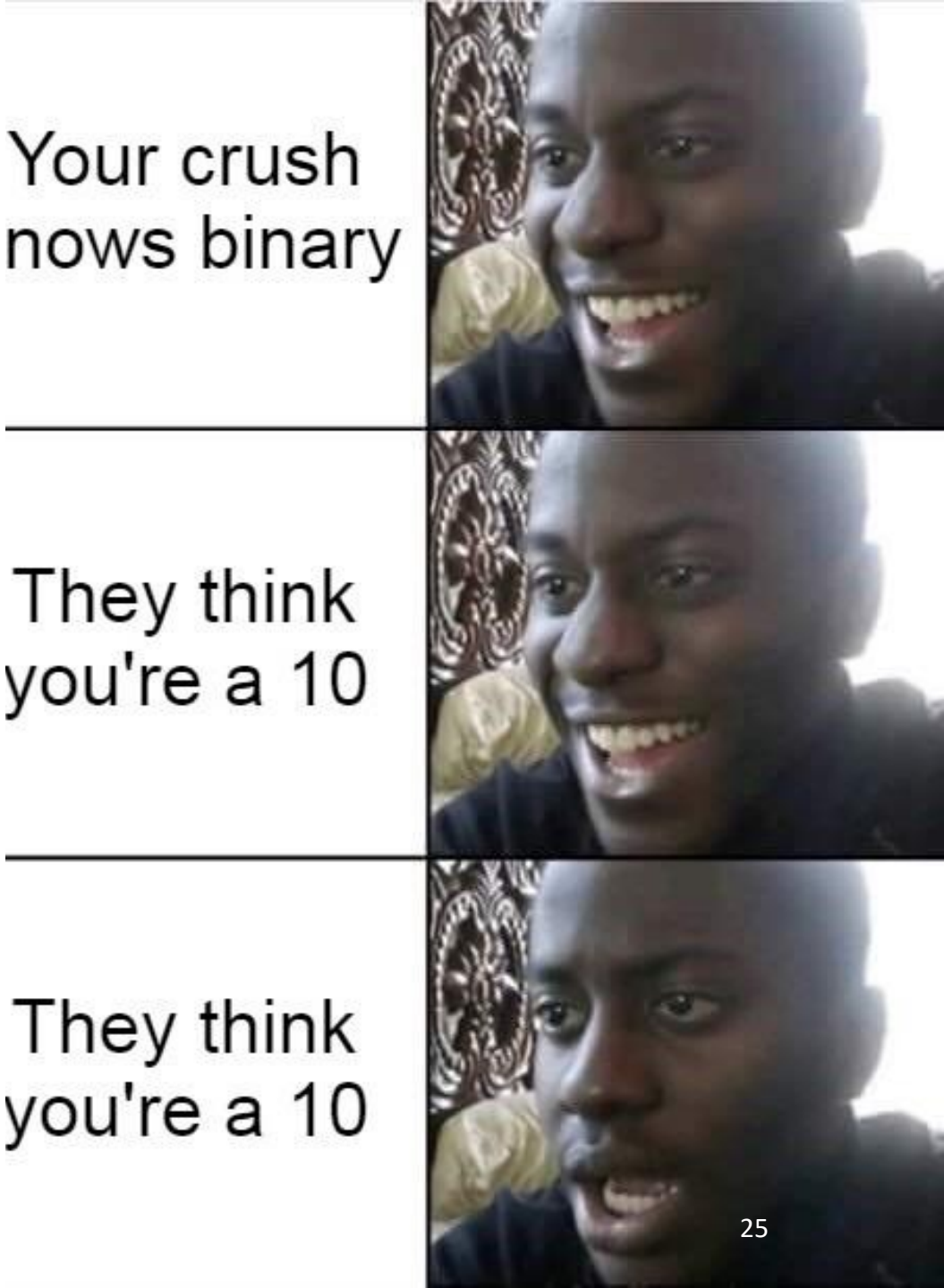
Associatività

- Per i numeri reali vale la proprietà associativa:

$$(a+b)+c = a+(b+c)$$

- Viste le limitazioni di rappresentazione, non sempre la proprietà associativa vale nel calcolatore!
  - $a=-1.5*10^{38}$
  - $b= 1.5*10^{38}$
  - $c= 1.0$
- $(a+b)+c = (-1.5*10^{38} +1.5*10^{38})+1.0 = 0+1=1$
- $a+(b+c) = -1.5*10^{38} +(1.5*10^{38}+1.0) = -1.5*10^{38} + 1.5*10^{38} = 0$

Pausa
(5 min)


Your crush nows binary
They think you're a 10
They think you're a 10

Architettura di Von Neumann e Linguaggio Macchina

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

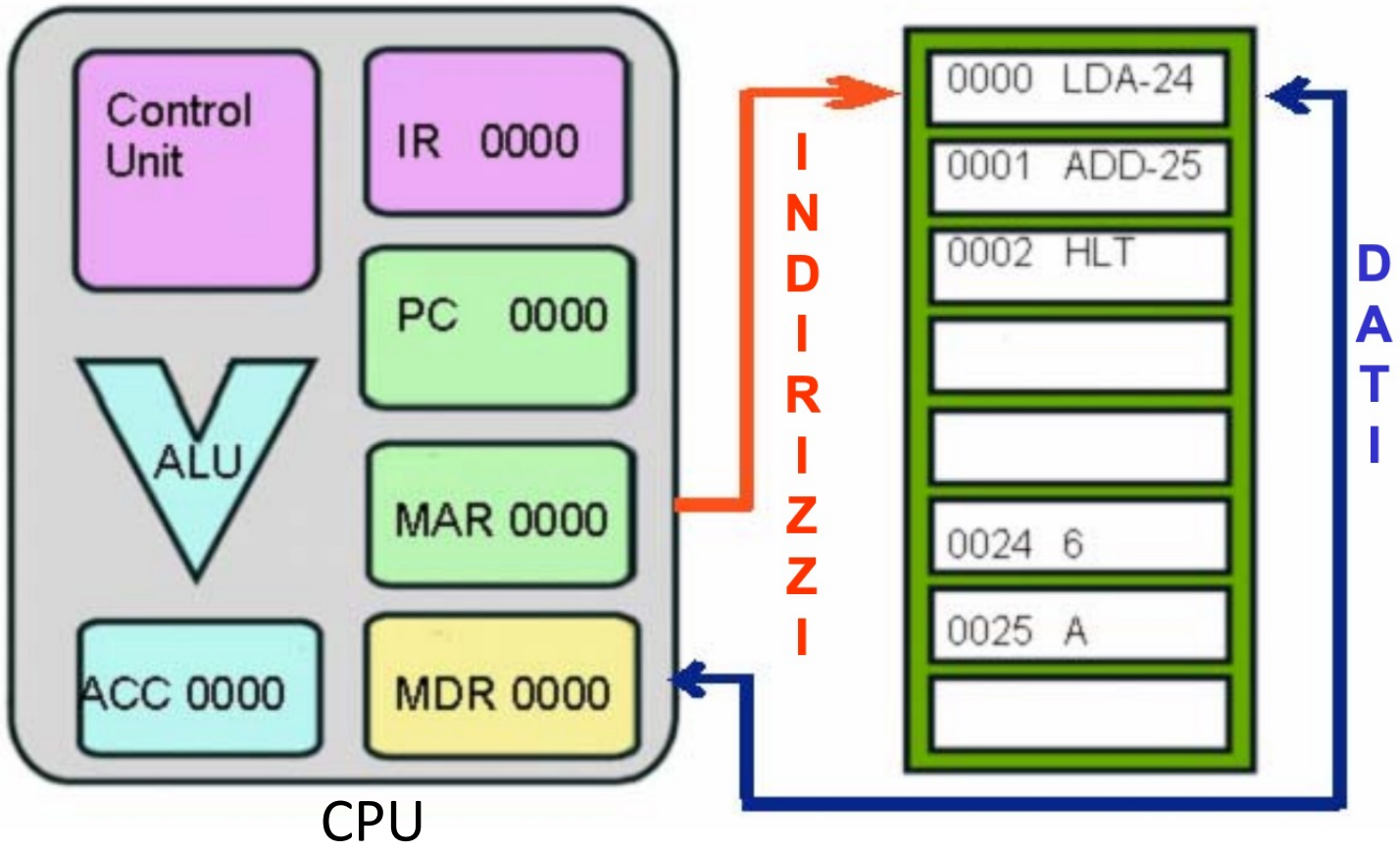
Linguaggio Macchina

- Il linguaggio della macchina è un linguaggio di basso livello (il livello di dettaglio tende a far perdere la visione d'insieme)
- Se cambia architettura, dobbiamo aggiornare il programma ☹
- Difficile da ricordare (sequenze di 0 e 1)
  - Soluzione: Il linguaggio assembler è una versione mnemonica del linguaggio macchina, es. ADD invece della sequeza di bit dell'addizione
  - l'assemblatore sostituisce le istruzioni con i corrispondenti codici macchina

Linguaggio Macchina

- Il Computer è capace di eseguire programmi in linguaggio macchina, tramite il ciclo FDE
  - [Fetch] - estrae una istruzione dalla memoria (in codice binario)
  - [Decode] – individua i circuiti corrispondenti all'istruzione
  - [Execute] - attiva tali circuiti

Architettura del Calcolatore

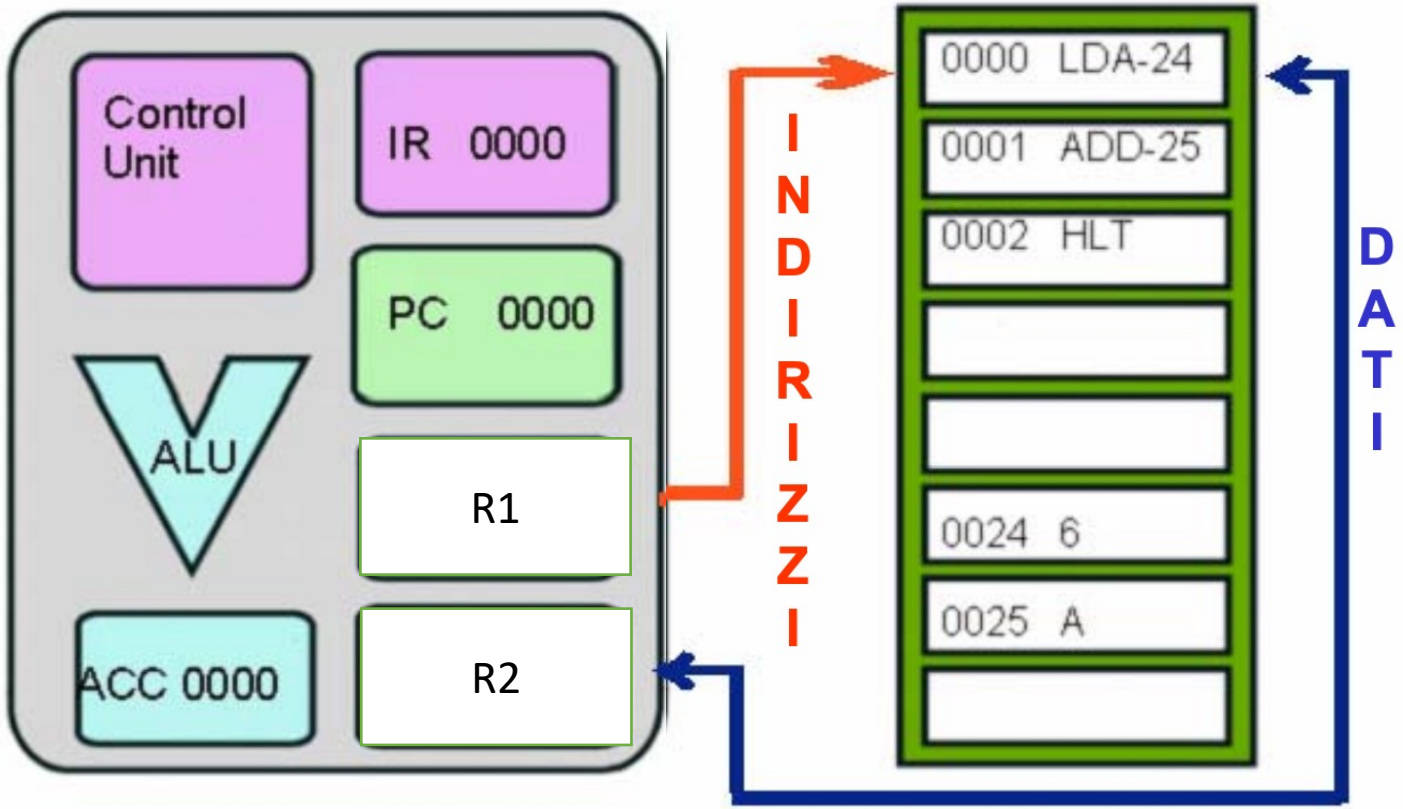


FDE=accesso/decodifica/esecuzione

Linguaggio Macchina

- Fetch
  - Recupera dalla memoria l'istruzione all'indirizzo indicato dal registro PC
  - Salva l'istruzione nel registro IR
  - aumenta il valore di PC di 4 (assumendo ogni istruzione occupi 4 byte), in modo da eseguire la prossima istruzione al turno successivo

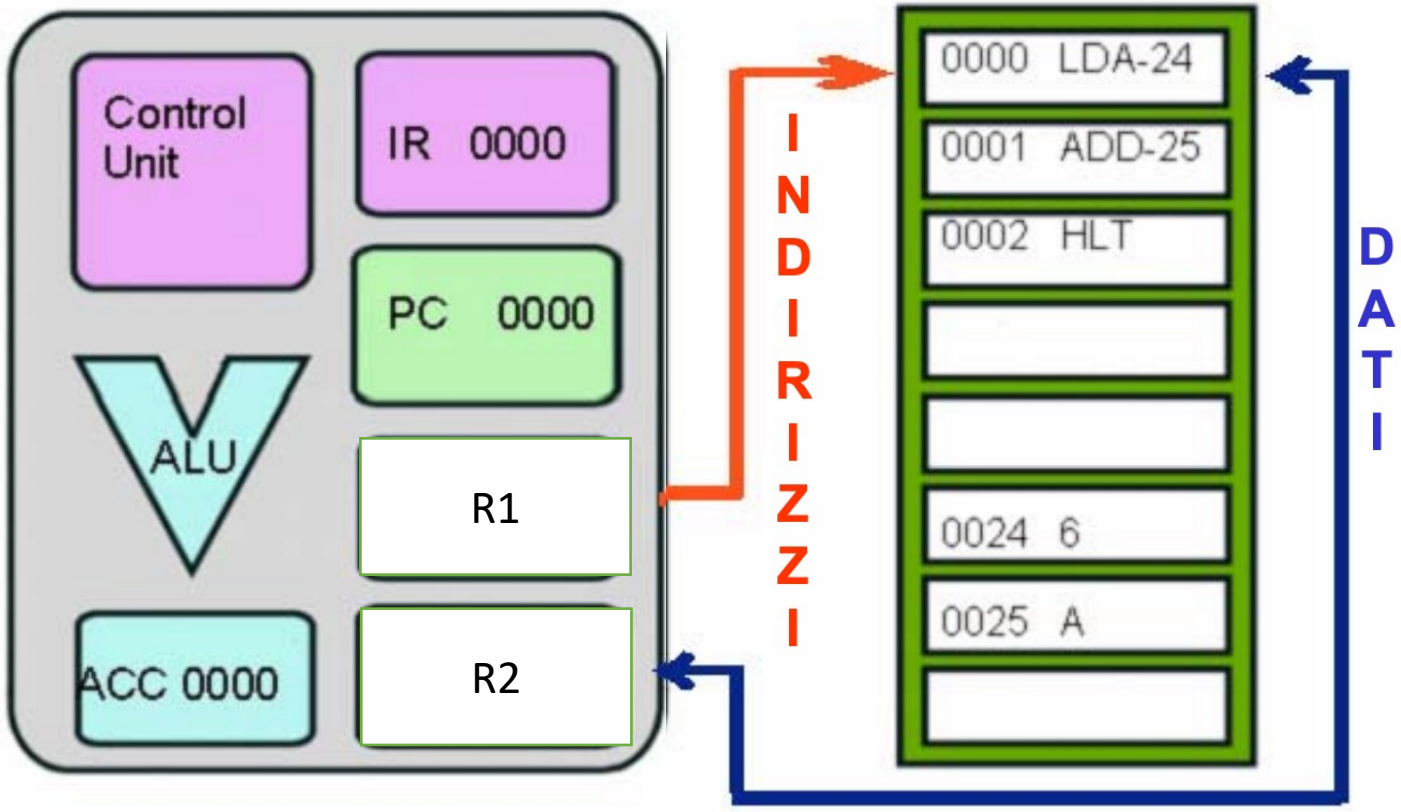
Architettura del Calcolatore



Linguaggio Macchina

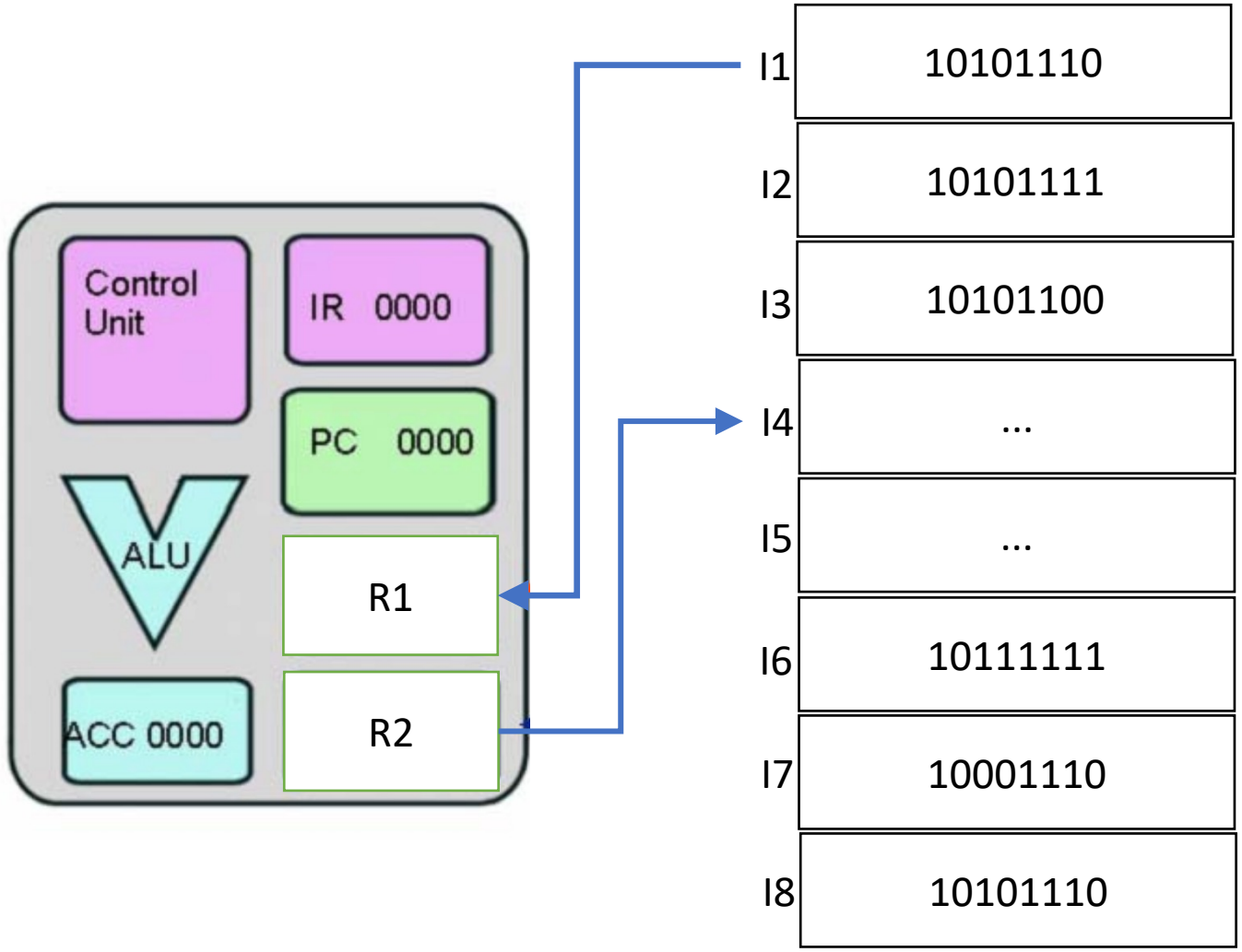
- Decode
  - Analizza il contenuto di IR per capire che istruzione deve eseguire
  - attiva i registri per l'input/output ed il corrispondente circuito della ALU o dell'unità di controllo
  - Es. se operazione aritmetico logica: attiva il circuito corrispondente nella ALU, collega i registri con l'input e quello con l'output
- Execute
  - attiva tali circuiti

Architettura del Calcolatore



Alcune Istruzioni

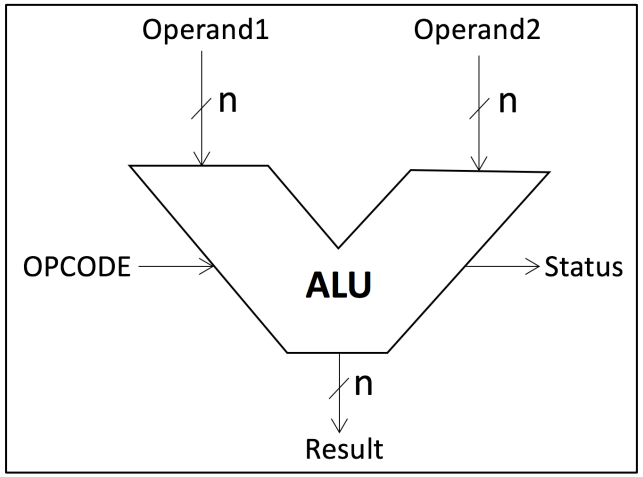
- I dati del programma, quelli che usiamo per fare i calcoli, stanno nella memoria RAM, le operazioni aritmetico/logiche operano sui registri, dobbiamo poter **trasferire dati da/nella RAM**:
  - Load(I1, R1): copia il contenuto della cella di memoria con indirizzo “I1” nel registro R1
  - Store(R2,I4): copia il contenuto del registro R2 nella cella di memoria con indirizzo ”I4”



Alcune Istruzioni

- Alcune Operazioni aritmetico-logiche
  - SommaInt(R1, R2, R3): somma il contenuto degli interi in R1 e R2 e salva il risultato in R3
  - SommaReali(R1, R2, R3): equivalente di SommaInt per numeri reali
  - ConvertiAIntero(R1,R2): converte il float in R1 in intero e salva il risultato in R2
  - ConfrontaInteri(R1, R2): salva in R2 1 se R1>R2, 0 R1=R2, -1 se R1<R2

- Le operazioni hanno al massimo due argomenti;
- Esistono operazioni diverse per interi e reali

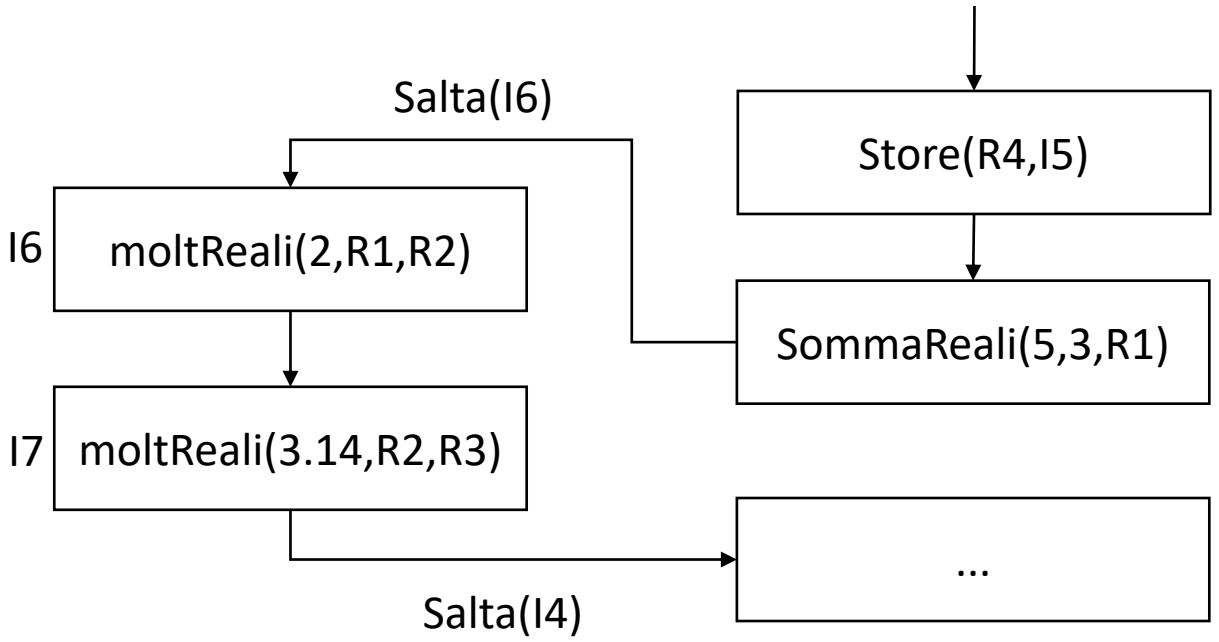


https://ati.ttu.ee/IAY0340/labs/Tutorials/SystemC/ALU.html

Istruzioni di Salto

- Salta(Ind): la prossima istruzione da eseguire sarà quella memorizzata in Ind
- SaltaSeUguale (R1, R2, Ind): se il contenuto di R1 e R2 sono uguali, la prossima istruzione da eseguire sarà quella memorizzata in Ind

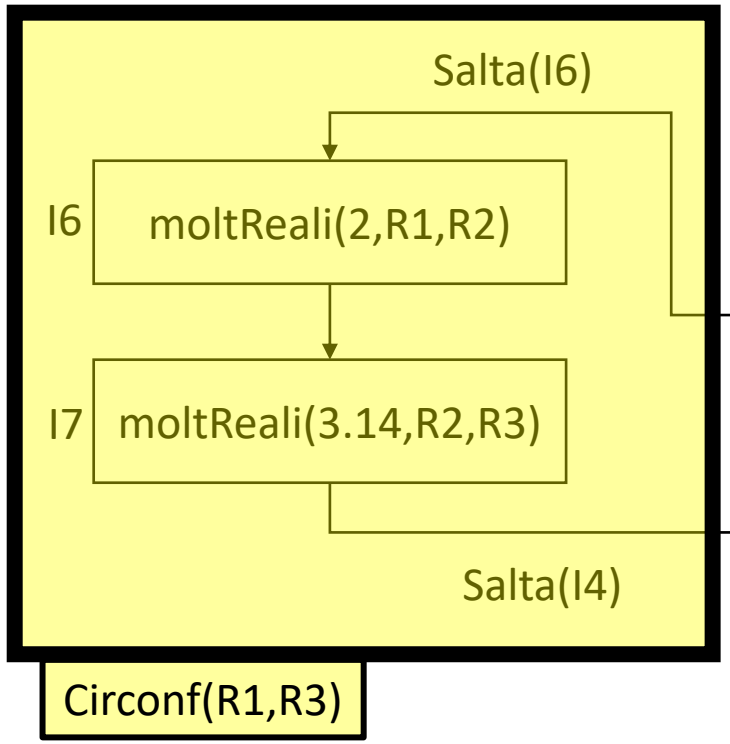
- Possiamo rappresentare in modo astratto istruzioni in linguaggio macchina tramite Diagramma di flusso

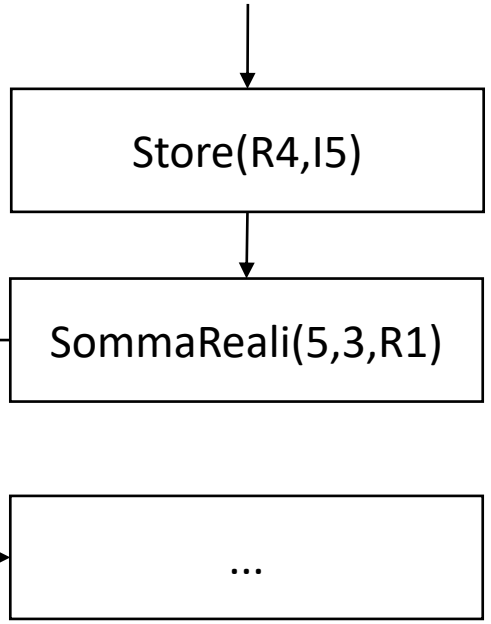


I1	Store(R4,I5)
I2	SommaReali(5,3,R1)
I3	Salta(I6)
I4	...
I5	...
I6	MoltReali(2,R1,R2)
I7	MoltReali(3.14,R2,R3)
I8	Salta(I2)

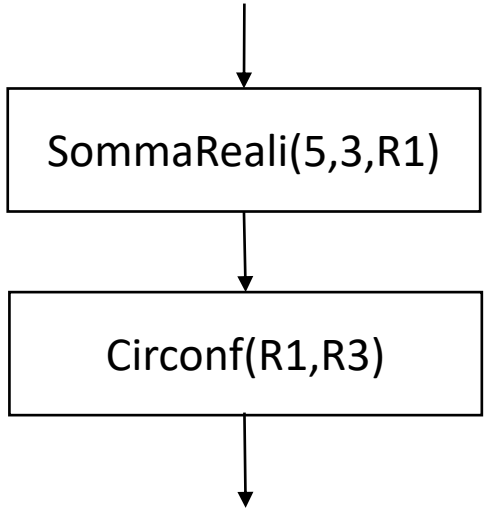
Diagrammi di Flusso

- Possiamo rappresentare in modo astratto istruzioni in linguaggio macchina tramite Diagramma di flusso





Si può saltare da un gruppo di istruzioni ad un altro modificando PC



Si può dare un nome a frammenti di codice e utilizzarli come nuovi comandi

I1	Store(R4,I5)
I2	SommaReali(5,3,R1)
I3	Salta(I6)
I4	...
I5	...
I6	MoltReali(2,R1,R2)
I7	MoltReali(3.14,R2,R3)
I8	Salta(I2)