

Materiale didattico di supporto alle lezioni del corso di

Fondamenti di Informatica

Corso di Laurea in Ingegneria Biomedica

Prof. Cinzia Bernardeschi

Dipartimento di Ingegneria dell'Informazione

Anno Accademico 2021-2022

Libri di testo:

- P. Corsini, G. Frosini
*Note su organizzazione di un calcolatore
rappresentazione dell'informazione*
Edizioni ETS, Pisa, 2017.
Piazza Carrara 16-19, Pisa
- Andrea Domenici, Graziano Frosini
*Introduzione alla Programmazione ed
Elementi di Strutture Dati con il Linguaggio C++*
Milano: Franco Angeli.
(va bene dalla quinta edizione in poi)
Acquistabile presso le principali librerie
- Raccolta di lucidi delle lezioni, esercitazioni: scaricabili dal **sito del corso**

<http://www.iet.unipi.it/c.bernardeschi/FondamentiDiInformatica.html>



Rappresentazione dell'informazione.

Architettura del calcolatore. Linguaggio Assembler

Concetti di base della programmazione

Concetto di algoritmo. Il calcolatore come esecutore di algoritmi. Linguaggi di programmazione ad alto livello. Sintassi e semantica di un linguaggio di programmazione. Metodologie di programmazione strutturata. Principi fondamentali di progetto e sviluppo di semplici algoritmi.

Programmare in C++

Tipi fondamentali. Istruzioni semplici, strutturate e di salto. Funzioni. Ricorsione. Riferimenti e puntatori. Array. Strutture e unioni. Memoria libera. Lettura/Scrittura da file. Visibilità e collegamento. Algoritmi di ricerca e di ordinamento. Concetti di base delle librerie. Limitazioni dei tipi derivati. Il concetto di tipo di dato astratto: le classi

Progettare ed implementare strutture dati

Liste, Code, Pile, Alberi, Tabelle.

Rappresentazione dell'Informazione

L'informazione è qualcosa di astratto.
Per poterla manipolare bisogna rappresentarla.

In un calcolatore i vari tipi di informazione (testi, figure, numeri, musica,...) si rappresentano per mezzo di sequenze di bit (cifre binarie).

Bit è l'abbreviazione di Binary digIT, numero binario.

Il bit è l'unità di misura elementare dell'informazione, ma anche la base del sistema numerico utilizzato dai computer.
Può assumere soltanto due valori: 0 o 1.

Byte è l'unità di misura dell'informazione che corrisponde ad 8 bit.

Rappresentazione dell'Informazione

Quanta informazione può essere contenuta in una sequenza di n bit?

L'informazione corrisponde a tutte le possibili disposizioni con ripetizione di due oggetti (0 ed 1) in n caselle (gli n bit), ossia 2^n

Esempio: $n=2$.

00

01

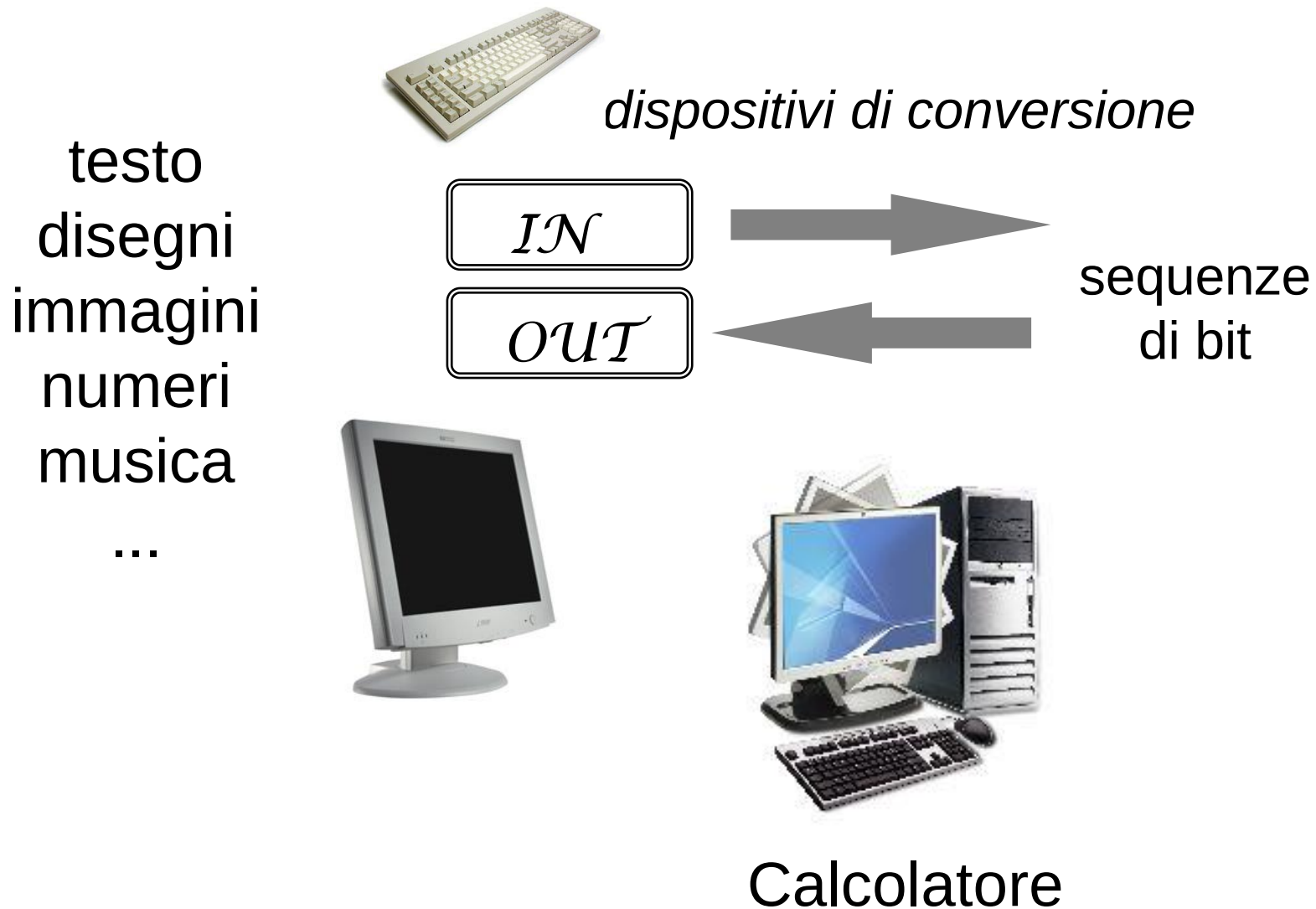
10

11

ATTENZIONE: Una stessa sequenza di bit può rappresentare informazione differente.

Per esempio 01000001 rappresenta

- l'intero 65
- il carattere 'A'
- il colore di un puntino sullo schermo



Rappresentazione del testo (1)

Codifica ASCII (American Standard Code for Information Interchange)
Standard su 7 bit (il primo bit del byte sempre 0)

Sequenze	Caratteri
00110000	0
00110001	1
00110010	2
...	...
00111001	9
...	...
01000001	A
01000010	B
01000011	C
...	...
01100001	a
01100010	b
...	...

01000011
01100001
01110010
01101111
00100000
01100001
01101110
01101001
01100011
01101111
00101100

Caro amico,

Rappresentazione del testo (2)

Codifica ASCII – Tabella di corrispondenza

3 cifre più significative

000	001	010	011	100	101	110	111	
NUL	DLE	SP	0	@	P	`	p	0000
SOH	XON	!	1	A	Q	a	q	0001
STX	DC2	"	2	B	R	b	r	0010
ETX	XOFF	#	3	C	S	c	s	0011
EQT	DC4	\$	4	D	T	d	t	0100
ENQ	NAK	%	5	E	U	e	u	0101
ACK	SYN	&	6	F	V	f	v	0110
BEL	ETB	'	7	G	W	g	w	0111
BS	CAN	(	8	H	X	h	x	1000
HT	EM	)	9	I	Y	i	y	1001
LF	SUB	*	:	J	Z	j	z	1010
VF	ESC	+	;	K	[	k	{	1011
FF	FS	,	<	L	\	l		1100
CR	GS	-	=	M	]	m	}	1101
SO	RS	.	>	N	^	n	~	1110
SI	US	/	?	O	_	o	DEL	1111

4 cifre meno
significative

Rappresentazione dei numeri naturali (1)

Base dieci

- Cifre: 0, 1, 2, 3, 4, 6, 7, 8, 9
- Rappresentazione posizionale

$(123)_{dieci}$ *significa* $1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

Base due

- Cifre: 0, 1
- Rappresentazione posizionale

$(11001)_{due}$ *significa* $1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$ ($= 25$)_{dieci}

Rappresentazione dei numeri naturali (2)

Data una base $\beta \geq \text{due}$

Ogni numero naturale N minore di β ($N < \beta$) è associato ad un simbolo elementare detto **cifra**

BASE	CIFRE
due	0 1
cinque	0 1 2 3 4
otto	0 1 2 3 4 5 6 7
sedici	0 1 2 3 4 5 6 7 8 9 A B C D E F

Rappresentazione dei numeri naturali (3)

I numeri naturali maggiori o uguali a β possono essere rappresentati da una sequenza di cifre secondo la *rappresentazione posizionale*

Se un numero naturale $N \geq \beta$ è rappresentato in base β dalla sequenza di cifre:

$$a_{p-1}a_{p-2} \dots a_1a_0$$

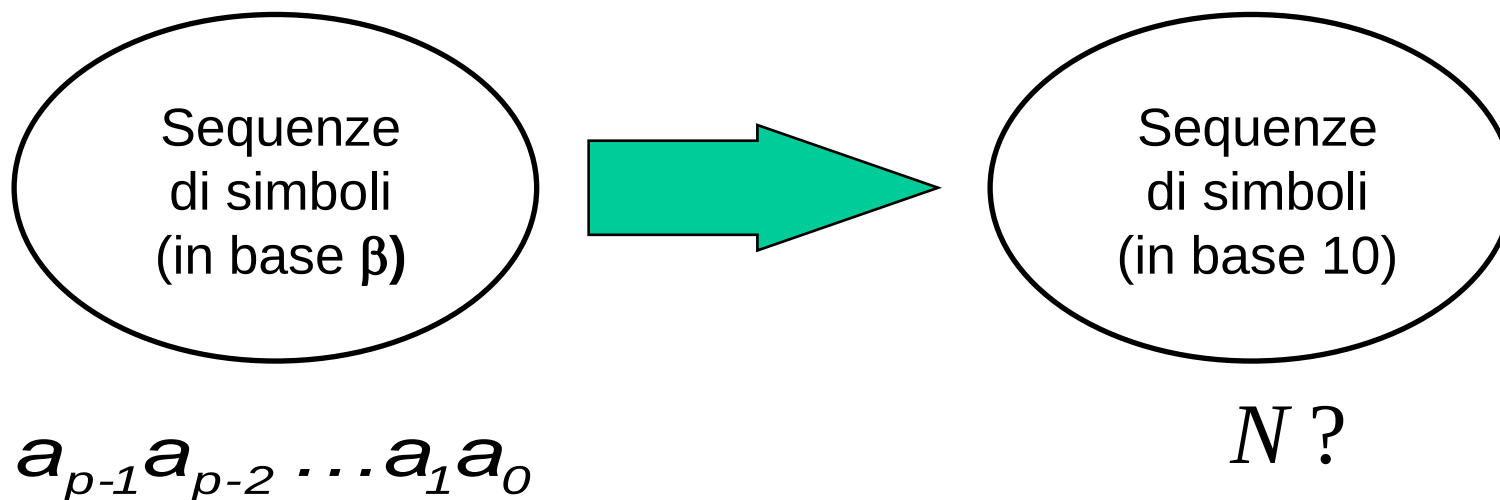
allora N può essere espresso come segue:

$$N = \sum_{i=0}^{p-1} a_i \beta^i = a_{p-1} \beta^{p-1} + a_{p-2} \beta^{p-2} + \dots + a_2 \beta^2 + a_1 \beta + a_0$$

dove a_0 è detta «cifra meno significativa» e a_{p-1} «cifra più significativa»

Rappresentazione dei numeri naturali (4)

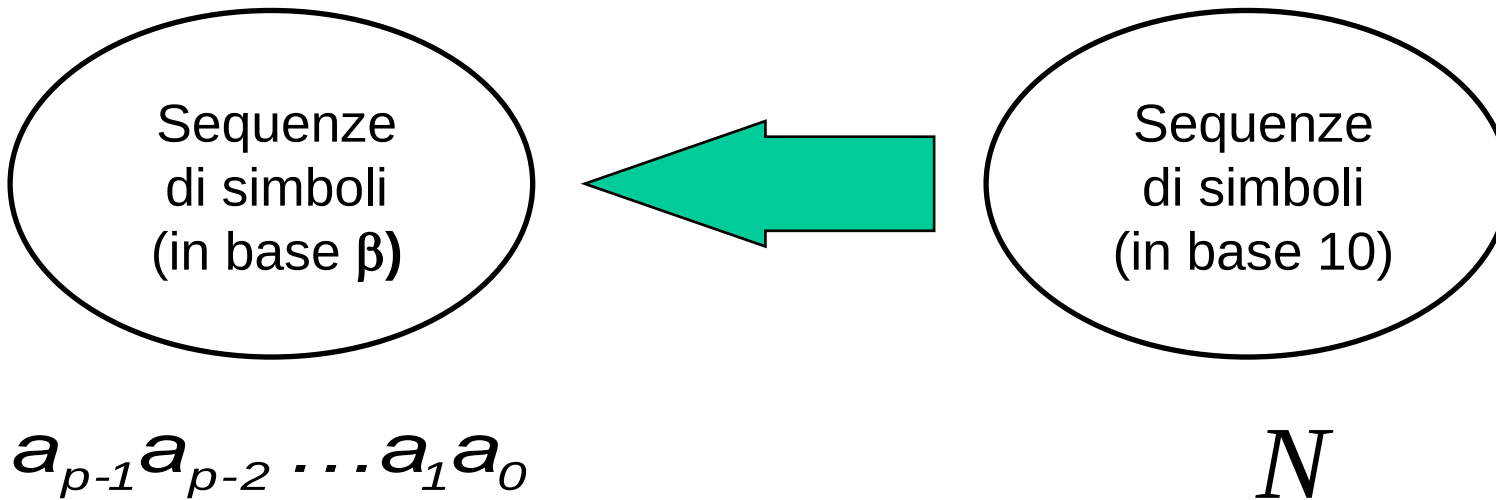
Data una sequenza di cifre in base β , a quale numero naturale corrisponde?



656_8	$\rightarrow$	$6 \cdot 8^2 + 5 \cdot 8^1 + 6 \cdot 8^0$	$\rightarrow$	430
$A03_{16}$		$10 \cdot 16^2 + 0 \cdot 16^1 + 3 \cdot 16^0$		2563
1101_2		$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$		13

Rappresentazione dei numeri naturali (5)

Data la base β ed un numero naturale N , trovare la sequenza di cifre che rappresenta N in base β



Rappresentazione dei numeri naturali (6)

Esempio: da base 10 a base 2

$$N = 23$$

inizio

	QUOZIENTE	RESTO	Rappresentazione
	23	-	
div 2	11	1	$11*2+1$
div 2	5	1	$((5*2)+1)*2+1$
div 2	2	1	$((((2*2)+1)*2)+1)*2+1$
div 2	1	0	$(((((1*2)+0)*2)+1)*2)+1)*2+1$
div 2	0	1	$((((((0*2+1)*2+0)*2+1)*2+1)*2)+1)*2+1$

fine

$(10111)_{due}$
 $a_4 a_3 a_2 a_1 a_0$

che in base dieci vale $1*2^4+...+1=(23)_{dieci}$, cvd

Rappresentazione dei numeri naturali (7)

Sia **mod** il resto e **div** il quoziente della divisione intera

Procedimento mod/div

Se $N = 0$ porre $a_0 = 0$; $\Rightarrow$ fine

Altrimenti: porre $q_0 = N$ e poi eseguire la seguente procedura iterativa:

$$q_1 = q_0 \text{ div } \beta$$

$$a_0 = q_0 \text{ mod } \beta$$

$$q_2 = q_1 \text{ div } \beta$$

$$a_1 = q_1 \text{ mod } \beta$$

...

$$q_{p-1} = q_{p-2} \text{ div } \beta$$

$$a_{p-1} = q_{p-2} \text{ mod } \beta$$

fino a che q_p è uguale a 0;

Il procedimento si arresta quando $q_p = 0$ (più precisamente subito dopo aver calcolato a_{p-1}). p è proprio il numero di cifre necessario per rappresentare N in base β

Esempio:

$$23 \text{ div } 2 = 11$$

$$23 \text{ mod } 2 = 1$$

NB: Il risultato della **mod** è sempre una cifra valida in base β , perché restituisce sempre un numero fra 0 e $\beta-1$ (estremi inclusi).

Rappresentazione dei numeri naturali (8)

Inizio $q_0 = N$

QUOZIENTE	RESTO
$q_0 = N$	-
$q_1 = q_0 / \beta$	a_0
$q_2 = q_1 / \beta$	a_1
$q_3 = q_2 / \beta$	a_2
$q_4 = q_3 / \beta$	a_3
$q_5 = q_4 / \beta$	a_4
$q_6 = q_5 / \beta$	a_5
$q_7 = \mathbf{0}$	a_6

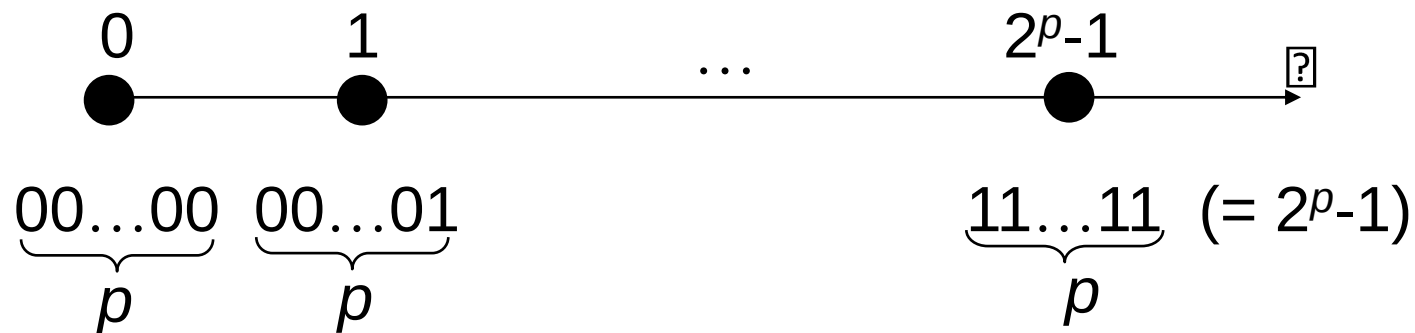
fine

$$N = a_6 \cdot \beta^6 + a_5 \cdot \beta^5 + a_4 \cdot \beta^4 + a_3 \cdot \beta^3 + a_2 \cdot \beta^2 + a_1 \cdot \beta^1 + a_0 \cdot \beta^0$$

Rappresentazione dei numeri naturali (9)

Potenza della base: $2^p \leftrightarrow (\underbrace{100\dots 00}_p)_{due}$

Intervallo di rappresentabilità con p bit



p	Intervallo $[0, 2^p-1]$
8	$[0, 255]$
16	$[0, 65535]$
32	$[0, 4294967295]$

NB: per la generica base β ,
l'intervallo di rappresentabilità
con p cifre è $[0, \beta^p-1]$

Rappresentazione dei numeri naturali (10)

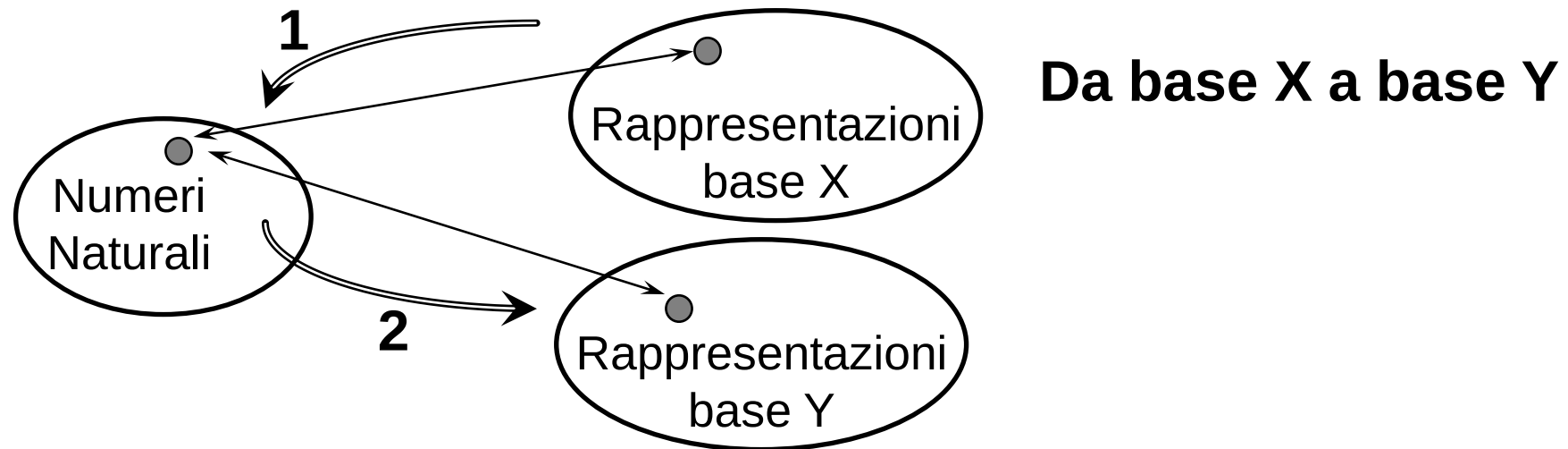
Calcolatore lavora con un numero finito di bit

- Supponiamo che $p = 16$ bit
- $A = 0111011011010101$ (30421)
 $B = 1010100001110111$ (43127)
- Poiché $A + B$ (73552) è maggiore di $2^p - 1$ (65535), quindi non è rappresentabile su p bit, si dice che la somma ha dato luogo ad **overflow**
- In generale, ci vogliono $p+1$ bit per la somma di due numeri di p bit

$A = 1111111111111111$ (65535)

$B = 1111111111111111$ (65535)

11111111111111110 (131070)



Trovare la rappresentazione in base 9 di $(1023)_{cinque}$

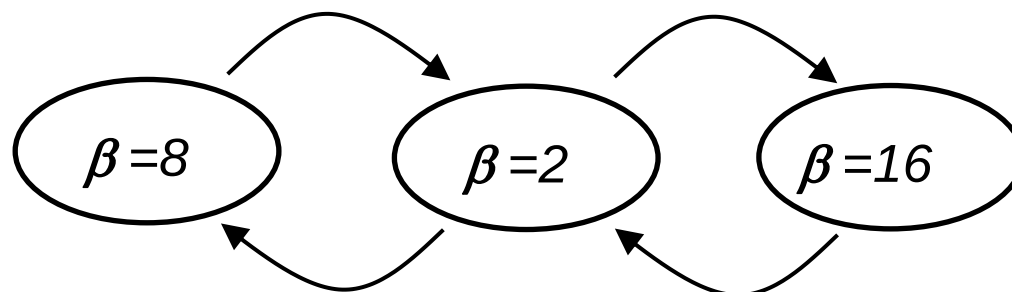
1) Trasformo in base 10 ed ottengo 138

2) Applico il procedimento `mod/div` ed ottengo $(163)_{nove}$

Casi particolari (potenze di 2)

$\beta = 8$

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111



$\beta = 16$

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

$(657)_{\text{otto}}$
 $\swarrow \quad \downarrow \quad \searrow$
 $(110 \ 101 \ 111)_{\text{due}}$

$(A03)_{\text{sedici}}$
 $\swarrow \quad \downarrow \quad \searrow$
 $(1010 \ 0000 \ 0011)_{\text{due}}$

$(FFE4)_{\text{sedici}}$
 $\swarrow \quad \swarrow \quad \downarrow \quad \searrow$
 $(1111 \ 1111 \ 1110 \ 0100)_{\text{due}}$

Modulo e segno

X rappresentazione di x

$|x|$ modulo di x

Intervallo di rappresentabilità con p bit
(doppia rappresentazione dello 0)

$$[-(2^{p-1}-1), +2^{p-1}-1]$$

- $p = 4$ $[-7, +7]$
- $p = 8$ $[-127, +127]$
- $p = 16$ $[-32767, +32767]$

$$X = \begin{cases} |x| & \text{se } x \geq 0 \\ 2^{p-1} + |x| & \text{se } x \leq 0 \end{cases}$$

$$x = \begin{cases} X & \text{se } X < 2^{p-1} \\ -(X - 2^{p-1}) & \text{se } X \geq 2^{p-1} \end{cases}$$

Ad esempio, per $p = 4$ si ha:

se $x = +1$ in base dieci

allora $X = 0001$

se $x = -6$ in base dieci

allora $X = 1110$

se $X = 0111$

allora $x = +7$ in base dieci

se $X = 1000$

allora $x = -0$ in base dieci

Complemento a 1

X rappresentazione di **x**

Intervallo di rappresentabilità con p bit
 $[-2^{p-1}-1, +2^{p-1}-1]$

- $p = 4$ $[-7, +7]$
- $p = 8$ $[-127, +127]$
- $p = 16$ $[-32767, +32767]$

$$X = \begin{cases} |x| & \text{se } x \geq 0 \\ 2^p - 1 - |x| & \text{se } x \leq 0 \end{cases}$$

$$x = \begin{cases} X & \text{se } X < 2^{p-1} \\ -(2^p - 1 - X) & \text{se } X \geq 2^{p-1} \end{cases}$$

Ad esempio, per $p = 4$ si ha:

se $x = +1$ in base dieci

allora $X = 0001$

se $x = -6$ in base dieci

allora $X = 1001$

se $X = 0111$

allora $x = +7$ in base dieci

se $X = 1000$

allora $x = -7$ in base dieci

Complemento a 2

X rappresentazione di **x**

Intervallo di rappresentabilità con p bit
 $[-2^{p-1}, +2^{p-1}-1]$

• $p = 4$ $[-8, +7]$

• $p = 8$ $[-128, +127]$

• $p = 16$ $[-32768, +32767]$

$$X = \begin{cases} |x| & \text{se } x \geq 0 \\ 2^p - |x| & \text{se } x < 0 \end{cases}$$

$$x = \begin{cases} X & \text{se } X < 2^{p-1} \\ -(2^p - X) & \text{se } X \geq 2^{p-1} \end{cases}$$

Ad esempio, per $p = 4$ si ha:

se $x = +1$ in base dieci

allora $X = 0001$

se $x = -6$ in base dieci

allora $X = 1010$

se $X = 0111$

allora $x = +7$ in base dieci

se $X = 1000$

allora $x = -8$ in base dieci

Numeri Interi (3)

x	X	$\{0,1\}^4$
0	0	0000
+1	1	0001
+2	2	0010
+3	3	0011
+4	4	0100
+5	5	0101
+6	6	0110
+7	7	0111
-8	8	1000
-7	9	1001
-6	10	1010
-5	11	1011
-4	12	1100
-3	13	1101
-2	14	1110
-1	15	1111

Rappresentazione di interi in complemento a due su calcolatore con $p=4$ bit

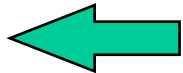
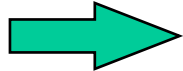
Anche in questo caso:

*numero negativo $\Leftrightarrow$ bit più significativo
della rappresentazione uguale a 1*

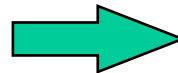
Inoltre, a differenza della
rappresentazione in modulo e segno,
non viene sprecata nessuna
rappresentazione (lo zero è
rappresentato una volta sola)

Numeri Interi (4)

$$\begin{array}{r} -2 \\ +1 \\ \hline -1 \end{array}$$

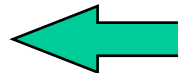


*compl.
a 2*



1110

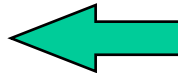
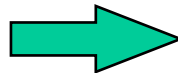
0001



1111

*Sommatore per
naturali*

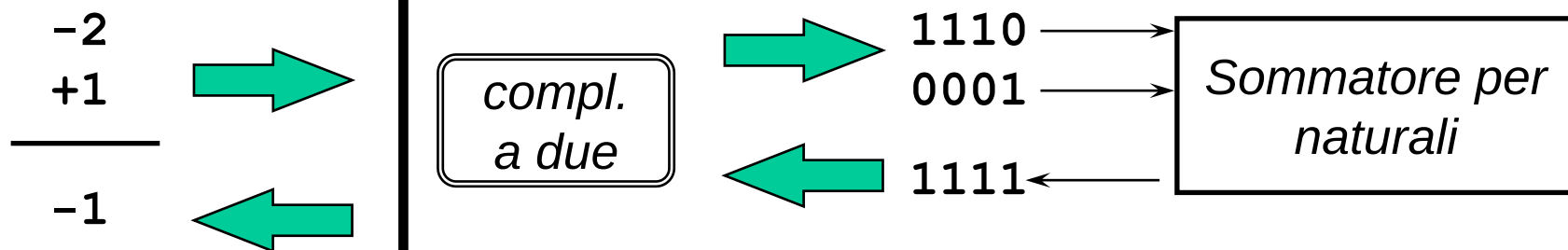
*Operazioni su
numeri*



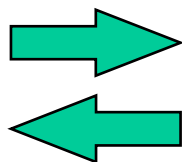
*Operazioni sulle
rappresentazioni*

*QUESTO è il motivo per cui i calcolatori
rappresentano gli interi in complemento a 2*

Numeri Interi (13/14)



Operazioni su
numeri



Operazioni sulle
rappresentazioni

QUESTO è il motivo per cui i calcolatori rappresentano gli interi in complemento a due: non occorre una nuova circuiteria per sommare e sottrarre numeri interi, viene utilizzata la stessa dei numeri naturali!

Sommando due numeri interi si verifica un **overflow** quando i due numeri hanno lo stesso segno ed il risultato ha segno diverso

Overflow

$$7+5=12$$

$$\begin{array}{r} 0111 + \\ 0101 \\ \hline \end{array}$$

$$\begin{array}{r} 1100 \Rightarrow -4 \end{array}$$

$$7-1=6$$

$$\begin{array}{r} 0111 + \\ 1111 \\ \hline \end{array}$$

$$\begin{array}{r} 10110 \Rightarrow 6 \end{array}$$

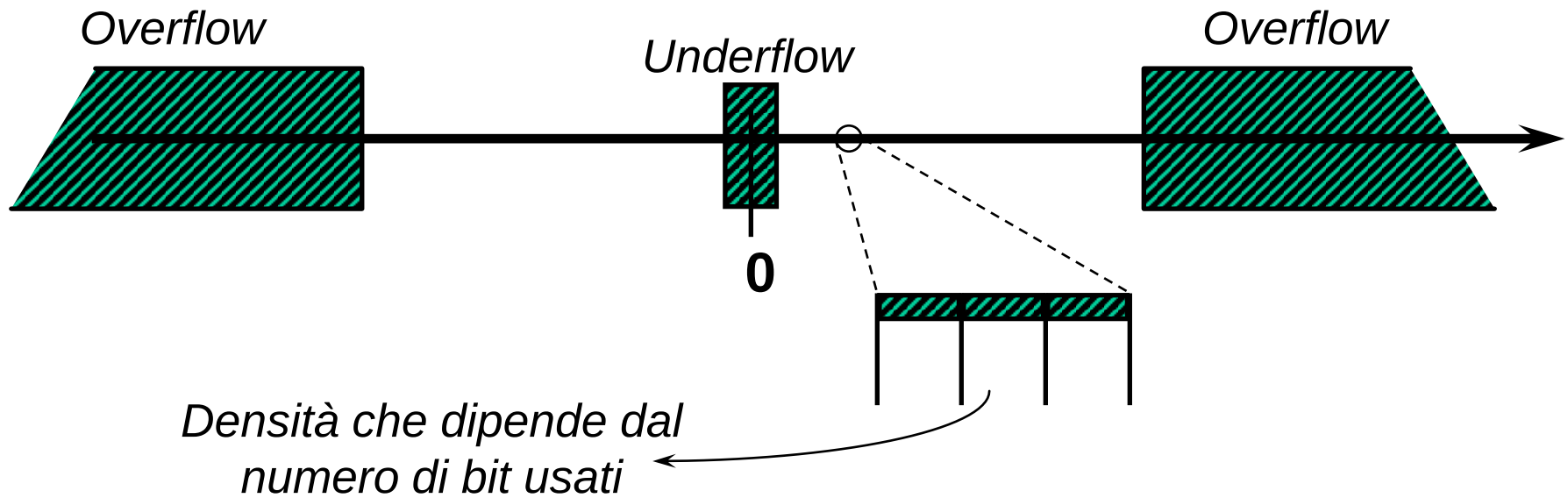
Ok

NB: L'1 più significativo viene scartato.
0110 su 4 bit è il risultato corretto cercato

Numeri Reali (1)

**Sottoinsieme
discreto** dei Numeri
Razionali

Sequenze
di bit



Numeri Reali – Virgola fissa (1)

- ❑ Si usa un numero fisso di bit per la parte intera ed un numero fisso di bit per la parte frazionaria.
- ❑ Sia r un numero reale da rappresentare
- ❑ Di seguito, indichiamo con $I(r)$ la parte intera e con $F(r)$ la parte frazionaria di r
- ❑ Siano p i bit utilizzati per rappresentare r e f i bit utilizzati per rappresentare la parte frazionaria $F(r)$ di r . Allora

$$R = a_{p-f-1} \dots a_0 a_{-1} \dots a_{-f} \quad \text{e} \quad r = a_{p-f-1} 2^{p-f-1} + \dots + a_0 2^0 + a_{-1} 2^{-1} \dots + a_{-f} 2^{-f}$$

- ❑ La virgola non si rappresenta.
- ❑ La parte intera $I(r)$ si rappresenta con le tecniche note.
- ❑ Per la parte frazionaria si usa il procedimento seguente:

Numeri Reali – Virgola fissa (2)

$$f_0 = F(r)$$

Se $f_0 \neq 0$ eseguire la seguente procedura iterativa:

$$a_{-1} = I(f_0 * 2) \qquad f_1 = F(f_0 * 2)$$

$$a_{-2} = I(f_1 * 2) \qquad f_2 = F(f_1 * 2)$$

...

fino a che f_j uguale a zero oppure si è raggiunta la precisione desiderata

Esempio:

$$p = 16 \text{ e } f = 5$$

$$r = +331,6875$$

Numeri Reali – Virgola fissa (3)

QUOZIENTE	RESTO
331	-
165	1
82	1
41	0
20	1
10	0
5	0
2	1
1	0
0	1



- Dalla rappresentazione dei numeri naturali: $331 \Leftrightarrow 101001011$;

Numeri Reali – Virgola fissa (4)

PRODOTTO	I	F
$0.6875 \times 2 \rightarrow 1.375$	1	0.375
$0.375 \times 2 \rightarrow 0.75$	0	0.75
$0.75 \times 2 \rightarrow 1.5$	1	0.5
$0.5 \times 2 \rightarrow 1$	1	0



- Parte frazionaria $\rightarrow 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} \rightarrow 0,5 + 0,125 + 0,0625 \rightarrow 0,6875$
- Dall'algoritmo precedente: $0,6875 \Leftrightarrow 0,1011$
- $r = (+101001011,1011)_{\text{due}}$
- $R = 0010100101110110$ (p=16, f=5)

ERRORE DI TRONCAMENTO

Rappresentare $x = 2.3$ con $f = 6$.

Per esprimere x sono necessarie un numero infinito di cifre:

$(10.0\underline{100110011001}\dots = 10.0\underline{1001}$ dove $\underline{1001}$ è la parte periodica)

Ne abbiamo a disposizione solo 6. Quindi si opera un troncamento alla 6^a cifra dopo la virgola.

Quindi, in realtà si rappresenta non x ma il numero x'

$$x' = 10.010011$$

$$x' = 2 + 2^{-2} + 2^{-5} + 2^{-6} = 2 + 0.25 + 0.03125 + 0.015625 = 2.296875$$

CALCOLO DI FISICA ASTRONOMICA

$m_e = 9 \times 10^{-28} \text{ gr}$; $m_{\text{sole}} = 2 \times 10^{+33} \text{ gr} \Rightarrow$ Intervallo di variazione $\approx 10^{+60}$.

Sarebbero quindi necessarie almeno 62 cifre, di cui 28 per la parte frazionaria.

Si hanno tante cifre perché l'intervallo da rappresentare è grande $\Rightarrow$ si separa l'intervallo di rappresentabilità dalla precisione, cioè dal numero di cifre.

Notazione Scientifica

$$r = \pm m \cdot \beta^e \quad m = \text{mantissa}; e = \text{esponente}$$

L'intervallo di rappresentabilità è fissato dal numero di cifre dell'esponente.

La precisione è fissata dal numero di cifre della mantissa.

Diverse rappresentazioni

Si fissa una forma standard

3.14

0.314 10^{+1}

31.4 10^{-1}

Notazione Scientifica

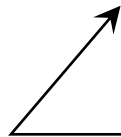
$$r = \pm m \cdot \beta^e \quad m = \text{mantissa}; e = \text{esponente}$$

L'intervallo di rappresentabilità è fissato dal numero di cifre dell'esponente.

La precisione è fissata dal numero di cifre della mantissa.

Forma Standard (unicità della rappresentazione)

3.14 0.314 10^{+1} 31.4 10^{-1}



Si fissa una forma standard.

IEEE 754-1985 (semplificata)

$$r \leftrightarrow \langle S, E, F \rangle$$

s = codifica del segno (1 bit)

F = numero naturale su M bit che codifica la mantissa

E = numero naturale su K bit che codifica l'esponente

$$r = s (1 + F 2^{-M}) * 2^E$$

forma normalizzata: $\pm (1 + F 2^{-M}) \cdot 2^{+E}$

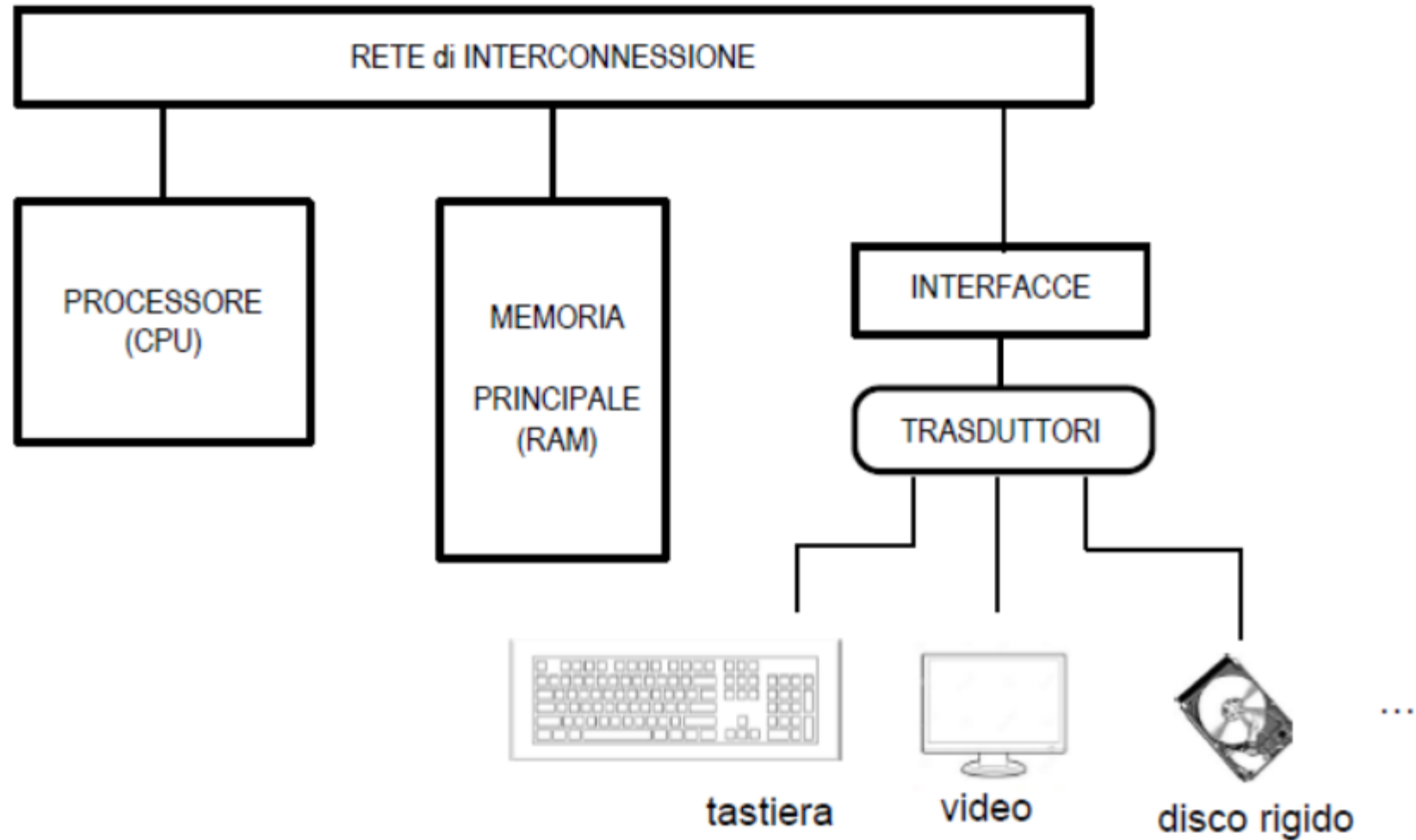
“uno implicito” $\Rightarrow$ lo zero non è rappresentabile

$p = 32$; $K = 8$; $M = 23$ (precisione semplice)

$p = 64$; $K = 11$; $M = 52$ (precisione doppia)

Schema a blocchi di un semplice calcolatore

Architettura di von Neumann (1946)



La memoria contiene dati e programmi (istruzioni) codificati in forma binaria

Il processore ripete all'infinito le azioni seguenti :

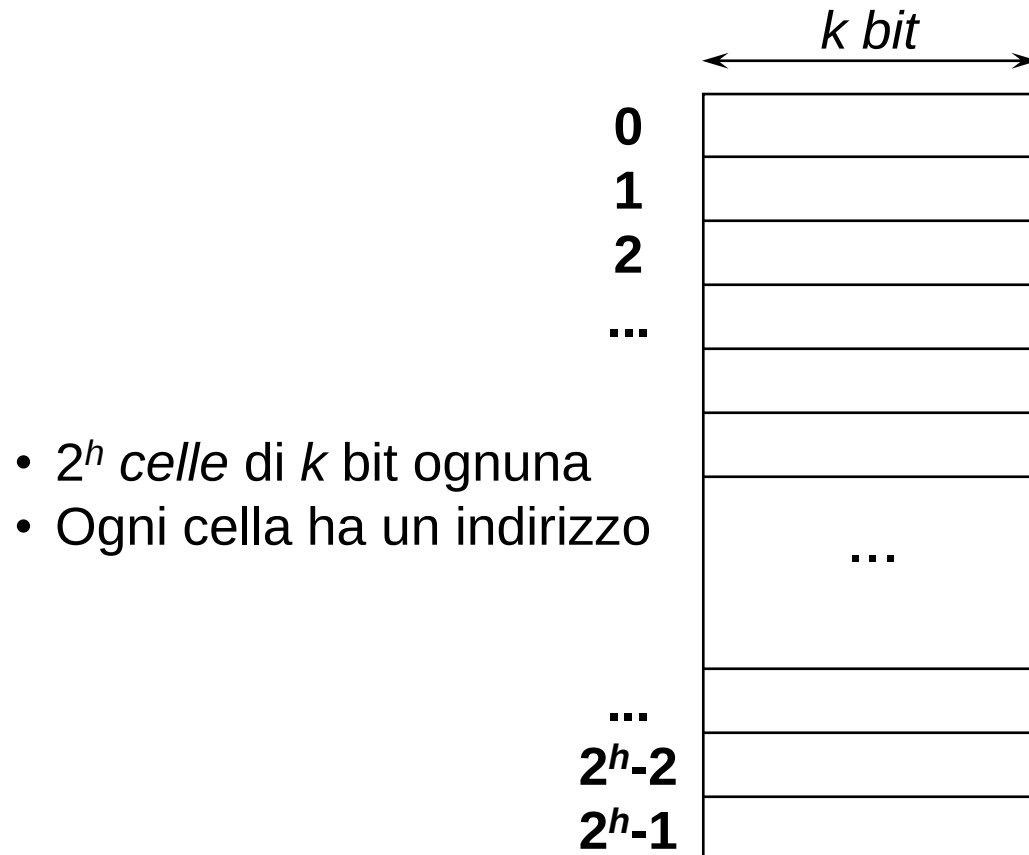
- preleva una nuova istruzione dalla memoria
- la decodifica
- la esegue

L'esecuzione di un'istruzione può comportare

- » Elaborazione e/o Trasferimento (memoria $\leftrightarrow$ processore, I/O $\leftrightarrow$ processore)

Le periferiche permettono al calcolatore di interagire con il mondo esterno

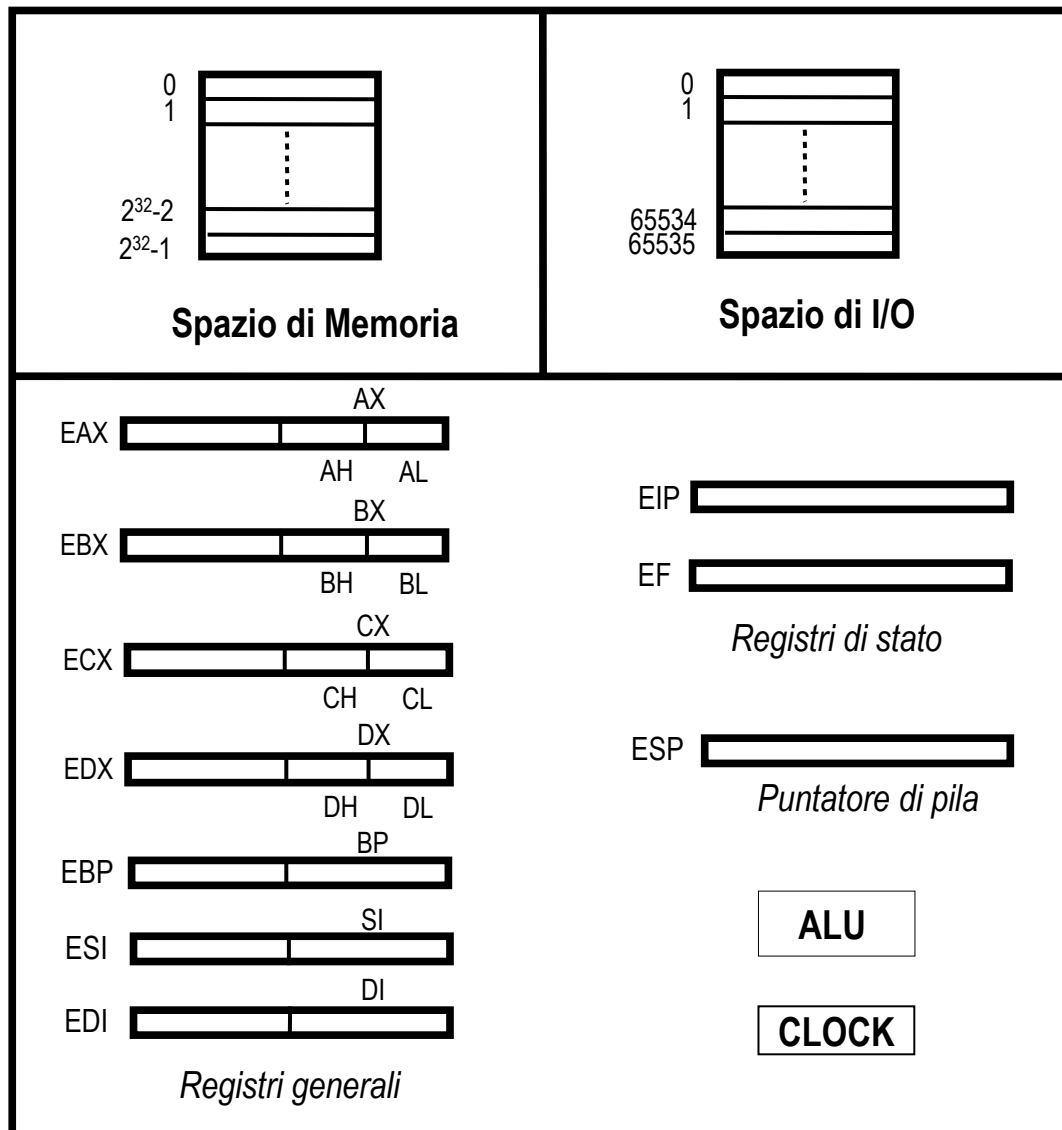
Struttura logica della memoria



OPERAZIONI

1. LETTURA di UNA cella
2. SCRITTURA di UNA cella

Struttura logica del processore (1)



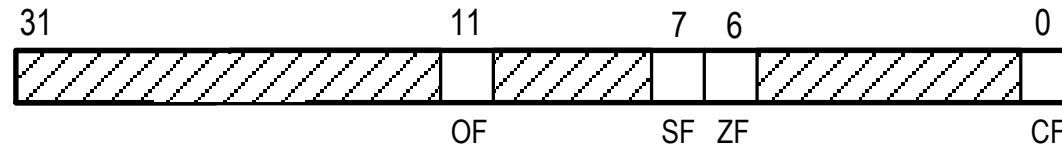
Struttura logica del processore (2)

Registro Extended Stack Pointer SSP

Registri di stato:

Extended Instruction Pointer EIP

Extended Flag register EF



Carry Flag CF

Zero Flag ZF

SignFlag SF

Overflow Flag OF

Formato delle istruzioni:

OPCODEsize source destination

OPCODEsize source

OPCODEsize destination

Dove **size** può essere:

B (byte): 8 bit

W (word): 16 bit

L (Long): 32 bit

Esempi:

MOVB (%EBX), %AL

MOVW (%EBX), %AX

MOVL (%EBX), %EAX

Istruzioni operative (1)

Istruzioni per il trasferimento di dati

MOV	Movimento (ricopiamento)
PUSH	Immissione di una word nella pila
POP	Estrazione di una word dalla pila
IN	Ingresso dati
OUT	Uscita dati

Istruzioni aritmetiche

ADD	Somma (fra interi oppure naturali)
SUB	Sottrazione (fra interi oppure naturali)
CMP	Confronto (fra interi oppure naturali)
MUL	Moltiplicazione (fra naturali)
DIV	Divisione (fra naturali)
IMUL	Moltiplicazione (fra interi)
IDIV	Divisione (fra interi)

Istruzioni operative (2)

Istruzioni logiche

NOT	Not logico bit a bit
AND	And logico bit a bit
OR	Or logico bit a bit

A = 11100000

B = 01100100

NOT A	00011111
A AND B	01100000
A OR B	11100100

Istruzioni di traslazione/rotazione

SHL	Traslazione a sinistra
SHR	Traslazione a destra
ROL	Rotazione a sinistra
ROR	Rotazione a destra

A = 11100000

SHL A	11000000
SHR A	01110000

ROL A	11000001
ROD A	01110000

Istruzioni di controllo

Istruzioni di salto

JMP	Salto incondizionato
Jcond	Salto sotto condizione

Istruzioni per la gestione dei sottoprogrammi

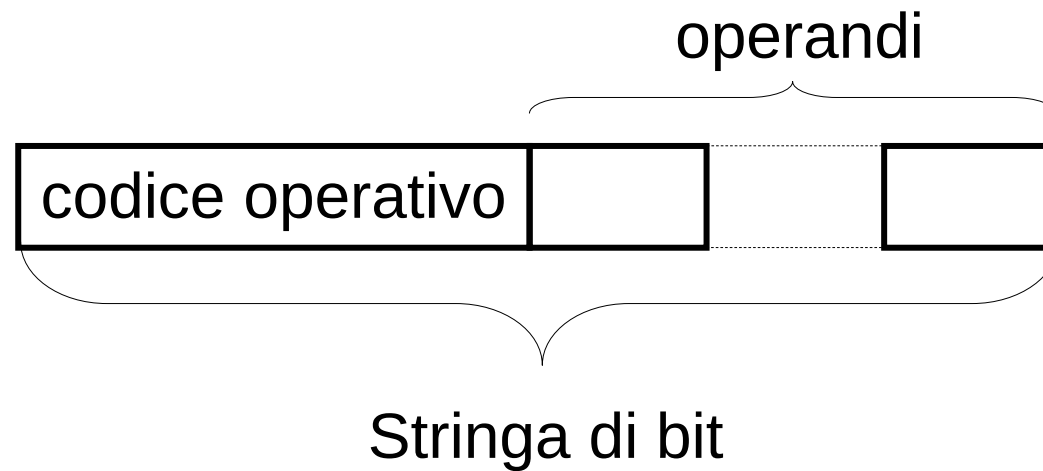
CALL	Chiamata di sottoprogramma
RET	Ritorno da sottoprogramma

Istruzione di alt

HLT	Alt
-----	-----

Formato delle istruzioni

Le istruzioni sono codificate come sequenze di bit.



Esempio

Codice Operativo	Operandi
0000	00110011 0001
MOV	immediato 51 %AL

Esempio di programma (1)

100	00000000	<i>Dato da elaborare (una WORD)</i>
102	00000000	<i>Risultato (un BYTE)</i>
103		
...		
...		
200	MOV \$0, %AL	<i>Azzera il contatore AL</i>
203	MOV (100), %DX	<i>Copia in DX il dato da elaborare</i>
207	CMP %DX, \$0	<i>Confronta il contenuto di DX con 0</i>
211	JE 232	<i>Salta se uguale</i>
215	SHL %DX	<i>Trasla a sinistra il contenuto di DX</i>
217	JC 225	<i>Salta se CF è settato all'indirizzo 225</i>
221	JMP 207	<i>Salta incondizionatamente a 207</i>
225	ADD \$1, %AL	<i>Incrementa il contatore AL</i>
228	JMP 207	<i>Salta incondizionatamente</i>
232	MOV %AL, (102)	<i>Memorizza il risultato</i>
236	HLT	<i>ALT</i>
237		

Contare il numero di bit a 1 in un dato inserito da tastiera (2 byte) memorizzato all'indirizzo 100. Scrivere il risultato all'indirizzo 102

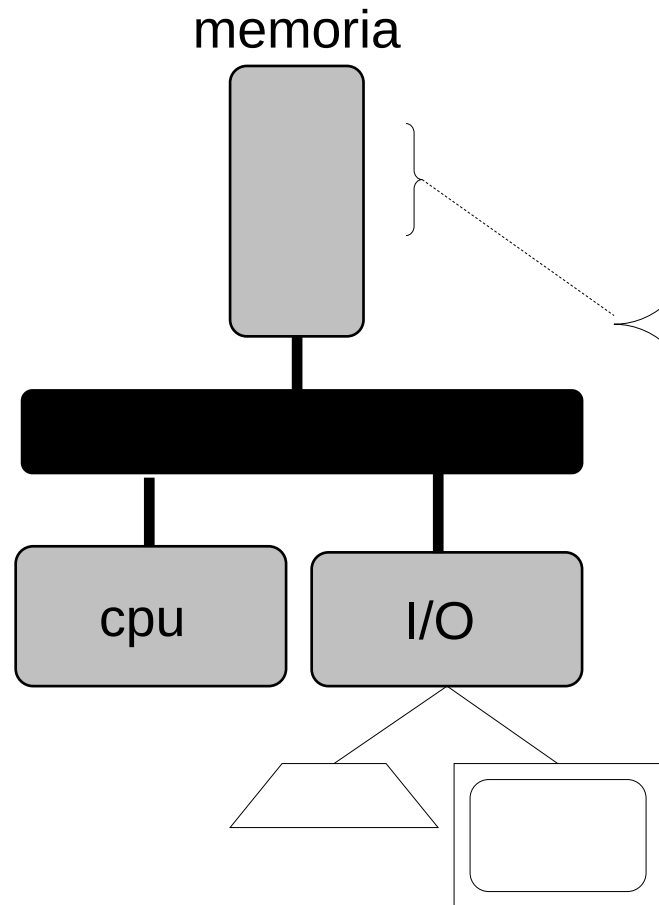
- Uso registro A per contare il numero di 1
- Salvo il dato nel registro D

esempio: D= 01000000000010001

Finchè il valore di D è diverso da 0

- Eseguo shift a sinistra di D
 - Se carry flag è settato (il bit più significativo era 1)
aggiungo 1 al registro A
-
- Scrivo A all'indirizzo 102 (AL: 00000011 3 bit uguali a 1)

Esempio di programma (2)



(100)

01000000

(101)

00010001

(102)

01000010

(200)

01000000

(201)

00010001

(202)

01000010

00010010

01100001

00010010

01010010

Livelli di astrazione dei Linguaggi

Esistono linguaggi a vari livelli di astrazione

Linguaggio macchina	sequenze di bit (difficile da leggere e capire) 0001001000110100
---------------------	---

Linguaggio Assembler	istruzioni macchina espresse con nomi simbolici (dipendente dalla macchina) MOV \$35, %AL
----------------------	---

Linguaggi ad Alto livello indipendenti dalla macchina	<pre>int main() { ... }</pre>
---	-----------------------------------

Semplice programma in C++

// Somma di due numeri interi inseriti da tastiera

```
int main()
{
    int a, b;
    cout << "immettere due numeri" << endl;
    cin >> a;
    cin >> b;
    int c = a + b;
    cout << "Somma: " << c << endl;
    return 0;
}
```

Informatica (definizione informale): è la scienza della rappresentazione e dell'elaborazione dell'informazione

Informatica (definizione formale dell'Association for Computing Machinery - ACM): è lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione, la loro teoria e analisi, il loro progetto, e la loro efficienza, realizzazione e applicazione.

Algoritmo: sequenza precisa e finita di operazioni, comprensibili e perciò eseguibili da uno strumento informatico, che portano alla realizzazione di un compito.

Esempi di algoritmi:

- Istruzioni di montaggio di un elettrodomestico
- Somma in colonna di due numeri
- Bancomat

Compito dell'esperto informatico: data la descrizione di un problema, produrre algoritmi (cioè capire la sequenza di passi che portano alla soluzione del problema) e codificarli in programmi.

- La descrizione di un problema non fornisce in generale un modo per risolverlo.
- La descrizione del problema deve essere chiara e completa.

Calcolatori Elettronici come esecutori di algoritmi: gli algoritmi vengono descritti tramite programmi, cioè sequenze di istruzioni scritte in un opportuno linguaggio comprensibile al calcolatore.

Algoritmo: sequenza precisa (non ambigua) e finita di operazioni, che portano alla realizzazione di un compito.

Le operazioni utilizzate appartengono ad una delle seguenti categorie:

1. **Operazioni sequenziali**

Realizzano una singola azione. Quando l'azione è terminata passano all'operazione successiva.

2. **Operazioni condizionali**

Controllano una condizione. In base al valore della condizione, selezionano l'operazione successiva da eseguire.

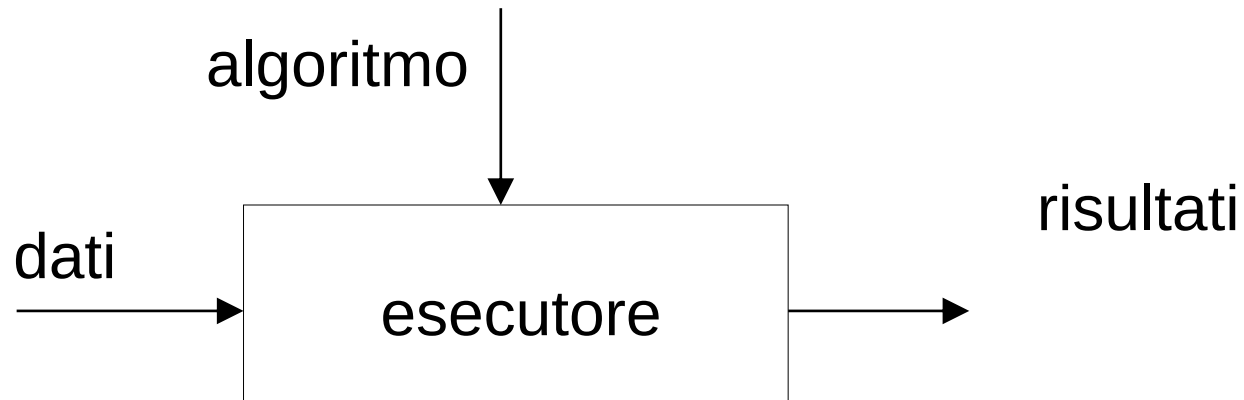
3. **Operazioni iterative**

Ripetono l'esecuzione di un blocco di operazioni, finchè non è verificata una determinata condizione.

Algoritmo (2)

L'esecuzione delle azioni nell'ordine specificato dall'algoritmo consente di risolvere il problema.

Risolvere il problema significa produrre risultati a partire da dati in ingresso



L'algoritmo deve essere applicabile ad un qualsiasi insieme di dati in ingresso appartenenti al dominio di definizione dell'algoritmo (se l'algoritmo si applica ai numeri interi deve essere corretto sia per gli interi positivi che per gli interi negativi)

Calcolo equazione $ax+b = 0$

- leggi i valori di a e di b
- calcola $-b$
- dividi quello che hai ottenuto per a e chiama x il risultato
- stampa x

Calcolo del massimo fra due numeri

- leggi i valori di a e di b
- **se** $a > b$ stampa a **altrimenti** stampa b

Calcolo del massimo di un insieme

- scegli un elemento come massimo provvisorio *max*
- per ogni elemento *i* dell'insieme:
 - se** $i > max$ eleggi *i* come nuovo massimo provvisorio *max*
- il risultato è *max*

Stabilire se una parola *P* precede alfabeticamente una parola *Q*.
Ipotesi: *P* e *Q* di uguale lunghezza $> = 1$

leggi *P* e *Q*; **inizializza** trovato a 0

- **ripeti finché** (trovato vale 0 e lettere non finite)
 - se** prima lettera di *P* < prima lettera di *Q*
 - allora** scrivi vero; trovato = 1;
 - altrimenti se** prima lettera di *P* > prima lettera di *Q*
 - allora** scrivi falso; trovato = 1;
 - altrimenti** togli da *P* e da *Q* la prima lettera
- **se** trovato vale 0 **allora** scrivi falso

Algoritmo (5)

Eseguibilità: ogni azione deve essere eseguibile dall'esecutore in un tempo finito

Non-ambiguità: ogni azione deve essere univocamente interpretabile dall'esecutore

Finitezza: il numero totale di azioni da eseguire, per ogni insieme di dati in ingresso, deve essere finito

Algoritmi equivalenti

- hanno lo stesso dominio di ingresso
- hanno lo stesso dominio di uscita
- in corrispondenza degli stessi valori del dominio di ingresso producono gli stessi valori del dominio di uscita

- Due algoritmi equivalenti
 - Forniscono lo stesso risultato, ma possono avere diversa efficienza e possono essere profondamente diversi

Esempio: calcolo del Massimo Comun Divisore (MCD) fra due interi M e N

Algoritmo 1

- Calcola l'insieme A dei divisori di M
- Calcola l'insieme B dei divisori di N
- Calcola l'insieme C dei divisori comuni
- il massimo comun divisore è il massimo divisore contenuto in C

Algoritmo 2 (di Euclide)

Se due numeri, m e n , sono divisibili per un terzo numero, x , allora anche la loro differenza è divisibile per x .

Per dimostrarla, si può utilizzare la proprietà distributiva.

Supponiamo $m > n$.

$$m = kx$$

$$n = hx$$

$$m - n = kx - hx = x(k - h)$$

Dunque si può dire che: **$\text{MCD}(m, n) = \text{MCD}((m - n), n)$**

Algoritmo

- **ripeti finché** ($M \neq N$):
 - **se** $M > N$, sostituisci a M il valore $M - N$
 - **altrimenti** sostituisci a N il valore $N - M$
- il massimo comun divisore corrisponde a M (o a N)

Proprietà essenziali degli algoritmi:

Correttezza:

- un algoritmo è corretto se esso perviene alla soluzione del compito cui è preposto, senza difettare in alcun passo fondamentale.

Efficienza:

- un algoritmo è efficiente se perviene alla soluzione del compito cui è preposto nel modo più veloce possibile, compatibilmente con la sua correttezza.

La formulazione testuale di un algoritmo in un linguaggio comprensibile ad un calcolatore è detta PROGRAMMA.

Ricapitolando, per risolvere un problema:

- Individuazione di un procedimento risolutivo
- Scomposizione del procedimento in un insieme ordinato di azioni –
ALGORITMO
- Rappresentazione dei dati e dell'algoritmo attraverso un formalismo comprensibile al calcolatore: LINGUAGGIO DI PROGRAMMAZIONE

Linguaggi di Programmazione (1)

Perché non usare direttamente il linguaggio naturale?

Il LINGUAGGIO NATURALE è un insieme di parole e di regole per combinare tali parole che sono usate e comprese da una comunità di persone

- non evita le ambiguità
- non si presta a descrivere processi computazionali automatizzabili

Occorre una nozione di linguaggio più precisa.

Un LINGUAGGIO di programmazione è una notazione formale che può essere usata per descrivere algoritmi.

Si può stabilire quali sono gli elementi linguistici primitivi, quali sono le frasi lecite e se una frase appartiene al linguaggio.

Un linguaggio è caratterizzato da:

SINTASSI - insieme di regole formali per la scrittura di programmi, che fissano le modalità per costruire frasi corrette nel linguaggio

SEMANTICA - insieme dei significati da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio

- a parole (poco precisa e ambigua)
- mediante azioni (semantica operativa)
- mediante funzioni matematiche (semantica denotazionale)
- mediante formule logiche (semantica assiomatica)

Alfabeto V (lessico)

- insieme dei simboli con cui si costruiscono le frasi

Universo linguistico V^*

- insieme di tutte le frasi (sequenze finite) di elementi di V

Linguaggio L su alfabeto V

- un sottoinsieme di V^*

Come definire il sottoinsieme di V^* che definisce il linguaggio?

Specificando in modo preciso la sintassi delle frasi del linguaggio TRAMITE una **grammatica formale**

Grammatiche

Grammatica $G = \langle V, VN, P, S \rangle$

V	insieme finito di simboli terminali
VN	insieme finito di simboli non terminali
P	insieme finito di regole di produzione
S	simbolo non terminale detto simbolo iniziale

Data una grammatica G , si dice Linguaggio LG generato da G l'insieme delle frasi di V

- Derivabili dal simbolo iniziale S
- Applicando le regole di produzione P

Le frasi di un linguaggio di programmazione vengono dette programmi di tale linguaggio.

Grammatica BNF (1)

GRAMMATICA BNF (Backus-Naur Form) è una grammatica le cui regole di produzione sono della forma

$X ::= A$

X simbolo non terminale

A sequenza di simboli (terminali e non terminali)

Una grammatica BNF definisce quindi un linguaggio sull'alfabeto terminale V mediante un meccanismo di derivazione (riscrittura)

A deriva da X se esiste una sequenza di derivazioni da X ad A

$X ::= A_1$

A_2

...

A_n

unica regola che indica l'**alternativa** fra $A_1, \dots, A_n$

Grammatica BNF (2)

$G = \langle V, VN, P, S \rangle$

$V = \{\text{lupo, canarino, bosco, cielo, mangia, vola, canta, ., il, lo}\}$

$VN = \{\text{frase, soggetto, verbo, articolo, nome}\}$

$S = \text{frase}$

Produzioni P

frase ::= **soggetto verbo .**

soggetto ::= **articolo nome**

articolo ::= il
lo

nome ::= lupo
canarino
bosco
cielo

verbo ::= mangia
vola
canta

Esempio: derivazione della frase
"il lupo mangia."

frase -> **soggetto verbo.**

-> **articolo nome verbo.**

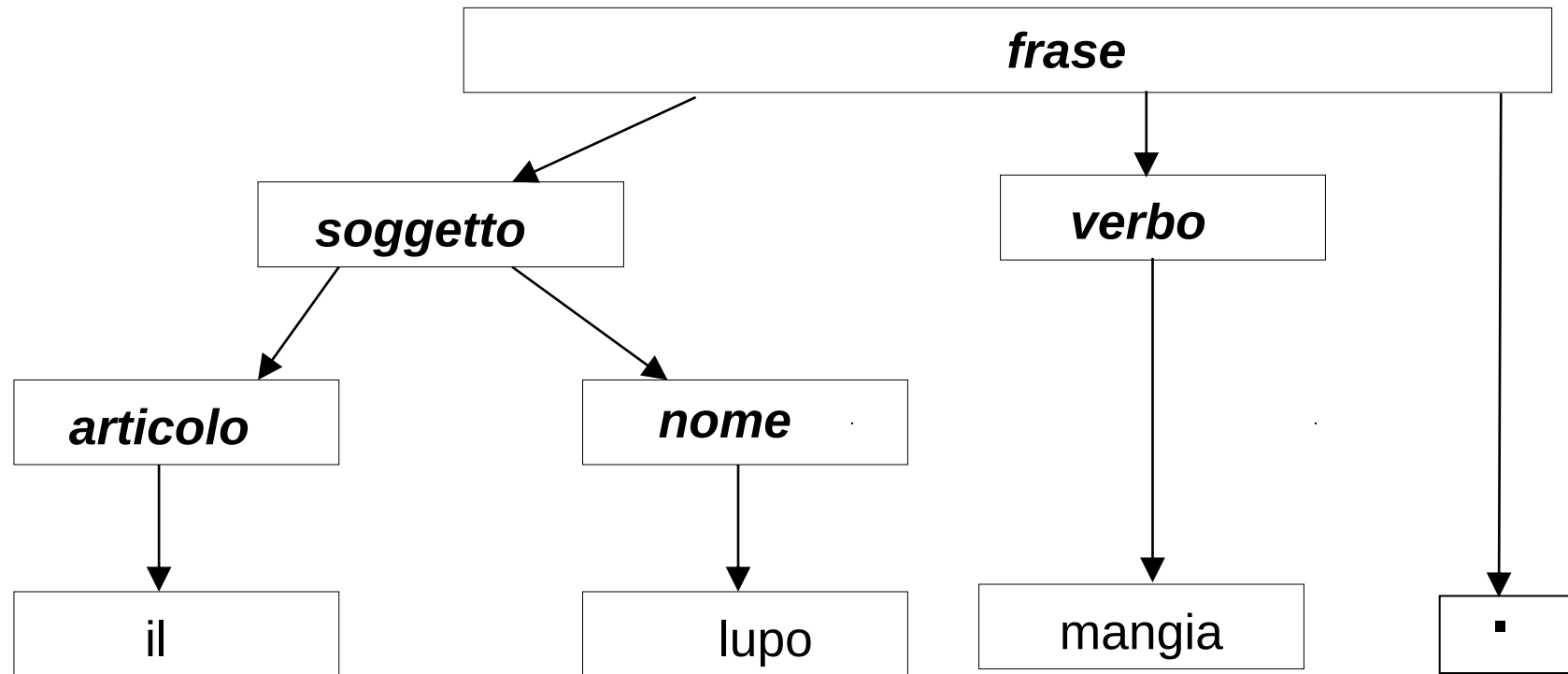
-> il **nome verbo.**

-> il lupo **verbo.**

-> il lupo mangia.

derivazione left-most

Albero sintattico



Albero sintattico: albero che esprime il processo di derivazione di una frase usando una data grammatica

Esistono programmi per generare analizzatori sintattici per linguaggi descritti con BNF

Sintassi e semantica

Scrivere un programma sintatticamente corretto non implica che il programma faccia quello per cui è stato scritto

La frase “il lupo vola” è sintatticamente corretta ma non è semanticamente corretta (non è significativa)

// Somma di due numeri interi inseriti da tastiera

```
int main()
{
    int a, b;
    cout << “immettere due numeri” << endl;
    cin >> a; cin >> b;
    int c = a + a; ←
    cout << “Somma: “ << c << endl;
    return 0;
}
```

Sviluppo di un programma (approccio compilato):

- editing: scrivere il testo e memorizzarlo su supporti di memoria permanenti
- compilazione
- linking
- esecuzione

Compilatore: traduce il programma sorgente in programma oggetto

- ANALISI programma sorgente
 - analisi lessicale
 - analisi sintattica
- TRADUZIONE
 - generazione del codice
 - ottimizzazione del codice

Esempi: C, C++, Pascal...

Sviluppo di un programma (approccio interpretato):

- editing: scrivere il testo e memorizzarlo su supporti di memoria permanenti
- interpretazione
 - ANALISI programma sorgente
 - analisi lessicale
 - analisi sintattica
 - ESECUZIONE

ESEMPIO: Basic, Java

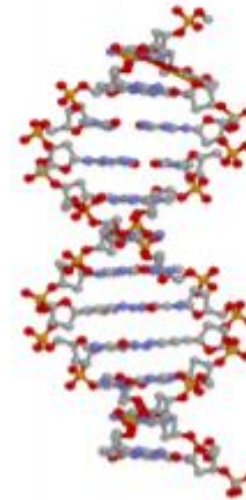
Esempi di applicazioni



Dispositivi medici



Chirurgia robotica



Bioinformatica

App medicali

