Argomenti del corso

Rappresentazione dell'informazione.
Architettura del calcolatore. Linguaggio Assembler

Concetti di base della programmazione
Concetto di algoritmo. Il calcolatore come esecutore di algoritmi. Linguaggi diprogrammazione 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. Visibilità e collegamento. Algoritmi di ricerca e di ordinamento.

Concetti di base della programmazione a oggetti
Limitazioni dei tipi derivati. Il concetto di tipo di dato astratto.

Programmare in C++
Classi. Operatori con oggetti classe. Altre proprietà delle classi. Classi per l'ingresso e per l'uscita.

Progettare ed implementare tipi di dato astratti
Alcuni tipi di dato comuni con le classi: Liste, Code, Pile.

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

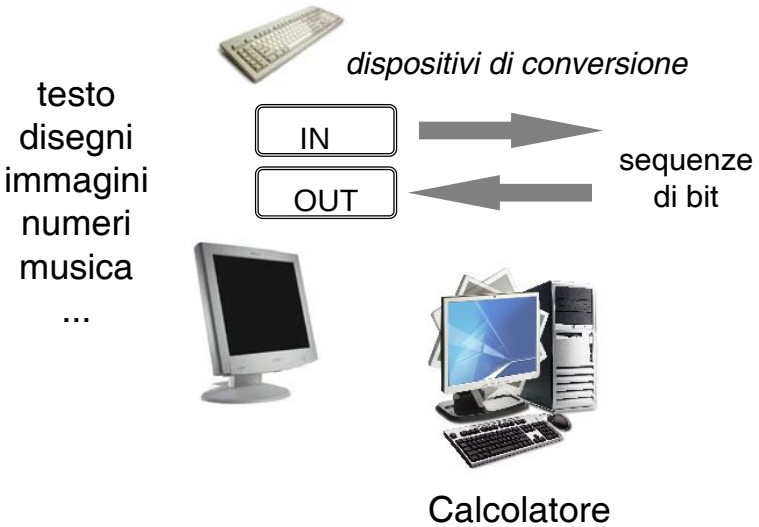
Quanta informazione può essere contenuta in una sequenza di n bit?
L'informazione corrisponde a tutte le possibili combinazioni degli n bit ossia $\mathbf{2^n}$

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

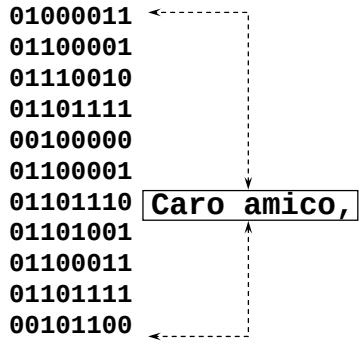
Calcolatore e Informazione



Rappresentazione del testo

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
...	...



Rappresentazione dei numeri naturali (1)

Base dieci

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

$(123)_{dieci}$ *significa* $1\times10^2 + 2\times10^1 + 3\times10^0$

Base due

- Cifre: 0, 1
- Rappresentazione posizionale

$(11001)_{due}$ *significa* $1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$ $(= 25)_{dieci}$

Rappresentazione dei numeri naturali (2)

Data una base $\beta \geq$ **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:

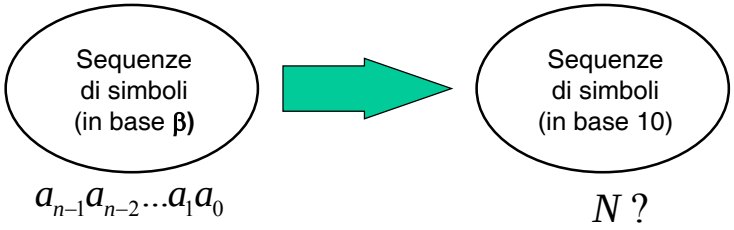
$$a_{n-1}a_{n-2}\ldots a_1a_0$$

allora *N* può essere espresso come segue:

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

Rappresentazione dei numeri naturali (4)

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



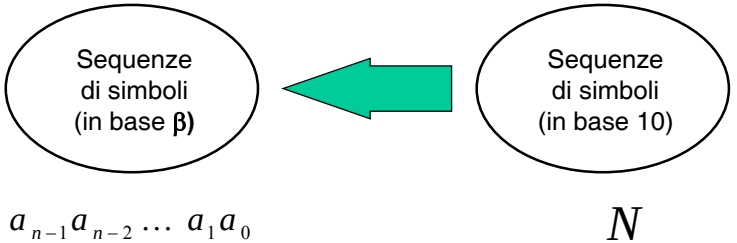
656_8 → $6 \cdot 8^2 + 5 \cdot 8^1 + 6 \cdot 8^0$ → 430

$A03_{16}$ → $10 \cdot 16^2 + 0 \cdot 16^1 + 3 \cdot 16^0$ → 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
	0	1	((((1*2)+0)*2)+1)*2)+1)*2+1

fine

$(10111)_{due}$

Rappresentazione dei numeri naturali (7)

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

Procedimento mod/div

q_0 = N
Se q_0 = 0 porre a_0 = 0;
altrimenti eseguire la seguente procedura iterativa:

$a_0 = q_0$ mod β $q_1 = q_0$ div β
$a_1 = q_1$ mod β $q_2 = q_1$ div β
...
fino a che q_p è uguale a 0;

Il procedimento si arresta quando $q_p = 0$, dove *p* è il numero di cifre necessario per rappresentare *N* in base β

Rappresentazione dei numeri naturali (8)

Inizio q_0 = N

	QUOZIENTE	RESTO
	q_0 = N	-
div β	$q_1 = q_0$ / ß	a_0
div β	$q_2 = q_1$ / ß	a_1
div β	$q_3 = q_2$ / ß	a_2
div β	$q_4 = q_3$ / ß	a_3
div β	$q_5 = q_4$ / ß	a_4
div β	$q_6 = q_5$ / ß	a_5
	q_7 = **0**	a_6

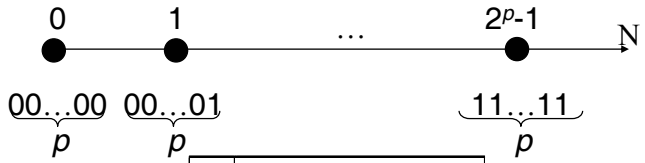
fine

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

Rappresentazione dei numeri naturali (9)

Potenza della base: $2^p \leftrightarrow (\underbrace{100\ldots00}_{p})_{due}$

Intervallo di rappresentabilità con *p* bit



p	Intervallo
8	[0, 255]
16	[0, 65535]
32	[0, 4294967295]

Rappresentazione dei numeri naturali (10)

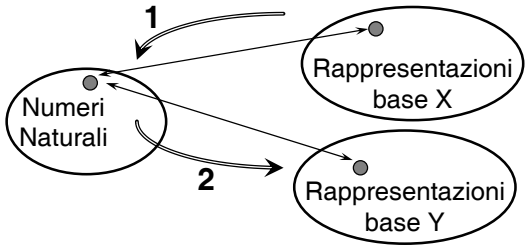
Calcolatore lavora con un numero finito di bit

- Supponiamo che *p* = 16 bit
- A = 0111011011010101 (30421)
  B = 1010100001110111 (43127)
- A + B è maggiore di 2^p-1(65535) quindi non è rappresentabile su *p* bit, allora 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)

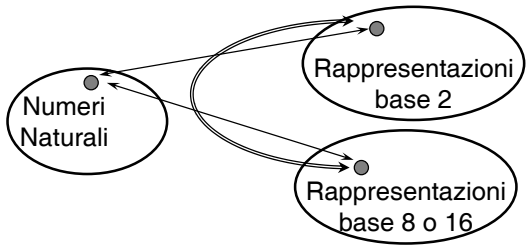
Numeri naturali - Cambio di base (1)



Da base X a base Y

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)



$(657)_{otto}$ $(A03)_{sedici}$ $(FFE4)_{sedici}$

$(\ 110\ 101\ 111\ )_{due}$ $(1010\ 0000\ 0011)_{due}$ $(1111\ 1111\ 1110\ 0100)_{due}$

Rappresentazione modulo e segno

Numero intero inteso come ±*modulo*

un bit — numero naturale

La notazione: (*segno_a*, *ABS_a*) rappresenta il numero intero

a = (*segno_a* == 0) ? +*ABS_a* : -*ABS_a*

Ad esempio, per $p = 4$ si ha:
(0, 011) rappresenta +*tre*
(1, 011) rappresenta -*tre*

(0, 000) rappresenta +*zero*
(1, 000) rappresenta –*zero*.

REGOLA: numero naturale A su p cifre ($a_{p-1}\ a_{p-2}\ \ldots.\ a_0$) in base *due* rappresenta il numero intero a in accordo alla relazione:

$$a = (a_{p-1} == 0)\ ?\ +A : -(A - due^{p-1})$$

A = 0101 rappresenta a = +0101 = +*cinque*
A = 1101 rappresenta a = -(1101 – 1000) = - 0101= –*cinque*

Rappresentazione modulo e segno

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

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

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

Rappresentazione complemento a due

Un numero naturale A su p bit in base *due* rappresenta il numero intero a (inteso come ±*modulo*) in accordo alla relazione:

$$a = (a_{p-1} == 0)\ ?\ +A : -(\sim A + 1)$$

dove ~A è il numero naturale ottenuto da A complementando tutti i bit di A

Si può dimostrare che $(A + \sim A) = due^p - 1$
Relazione che lega a ad A:

$$a = (a_{p-1} == 0)\ ?\ +A : -(due^p - A)$$

Ad esempio, per p = 4 si ha:
A = 0000 rappresenta a = +0000 = +*zero*
A = 0001 rappresenta a = +0001 = +*uno*

A = 0111 rappresenta a = +0111 = +*sette*
A = 1000 rappresenta a = -(0111 + 1) = -1000 = -*otto*
A = 1001 rappresenta a = -(0110 + 1) = -0111 = -*sette*
A = 1111 rappresenta a = -(0000 + 1) = -0001 = -*uno*

Numeri Interi (4)

Rappresentazione complemento a due

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]

Numeri Interi (5)

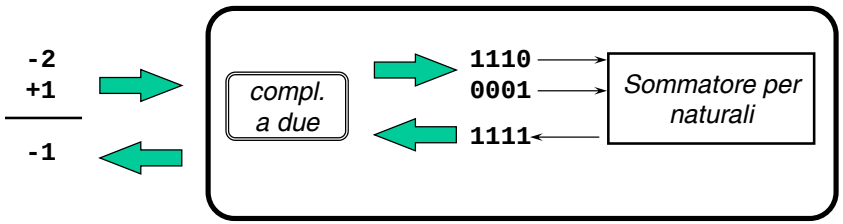
a	A	$\{0,1\}^4$
0	0	0000
+1	1	0001
+2	2	0010
+3	3	0011
+4	4	0100
+5	5	0101
+6	6	0110
+7	7	0111
-8	8	1000
-7	9	1001
-6	10	1010
-5	11	1011
-4	12	1100
-3	13	1101
-2	14	1110
-1	15	1111

Calcolatore con interi su 4 bit

Rappresentazione complemento a due

numero negativo ⇔ bit più significativo della rappresentazione uguale a 1

Numeri Interi(6)



Operazioni su numeri ⇄ *Operazioni sulle rappresentazioni*

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

Numeri Interi(7)

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

Overflow

Ok

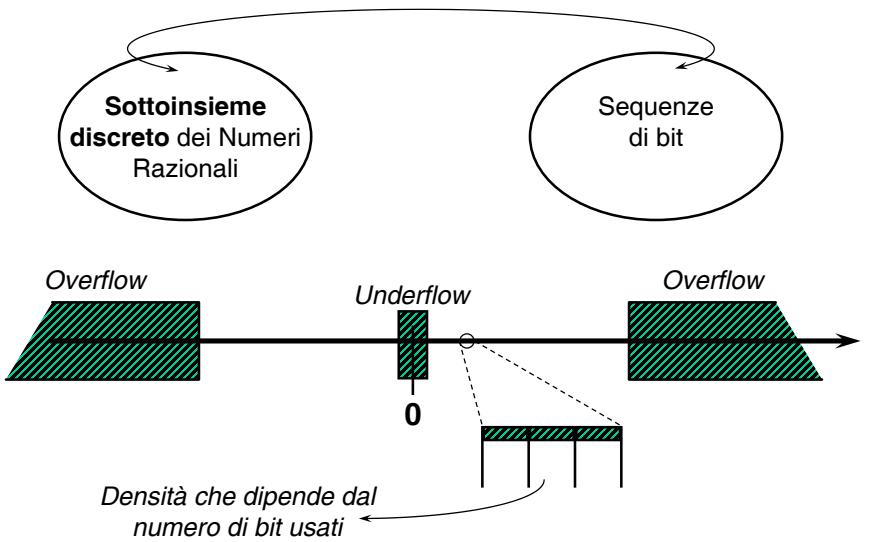
7+5=12

```
0111 +
0101
----------
1100   => -4
```

7-1=6

```
0111 +
1111
----------
1  0110  => 6
```

Numeri Reali



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 per rappresentare *r*: *f* per la parte frazionaria e (p-f) i bit la parte intera:

  $R = a_{p-f-1}\ldots a_0\ a_{-1}..a_{-f}$ $\quad r \cong a_{p-f-1}2^{p-f-1}+\ldots. + a_0 2^0 + a_{-1}2^{-1} \ldots + a_{-f}2^{-f}$

  Esempio: +1110.01 in base *due* vale +14.25 in base *dieci*
- 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) **f_1 = F(f_0 * 2)**
a_{-2} = I(f_1 * 2) **f_2 = F(f_1 * 2)**
...
fino a che f_j uguale a zero oppure si è raggiunta la precisione desiderata

Esempio:

***p* = 16 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 ⇔ 101001011;**

Numeri Reali – Virgola fissa(4)

PRODOTTO	I	F
0.6875*2->1.375	1	0.375
0.375*2 -> 0.75	0	0.75
0.75*2 -> 1.5	1	0.5
0.5*2 -> 1	1	0

- r = +10100101110110 ⇒ R = 0010100101110110
- Parte frazionaria ->

$1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 1*2^{-4}$ -> 0,5 + 0,125 + 0,0625 -> 0,6875

- Dall'algoritmo precedente: 0,6875 ⇔ 0,1011
- r = (+101001011,1011) base *due*

Numeri Reali – Virgola fissa(5)

ERRORE DI TRONCAMENTO

Rappresentare $r = 2.3$ con f = 6.

Per esprimere r sono necessarie un numero infinito di cifre.

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

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

$r' = 10.010011$

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

Numeri Reali – Virgola mobile(1)

CALCOLO DI FISICA ASTRONOMICA

$m_e = 9 \times 10^{-28}$ gr; $m_{sole} = 2 \times 10^{+33}$ gr ⇒ Intervallo di variazione ≈ 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 ⇒ si separa l'intervallo di rappresentabilità dalla precisione, cioè dal numero di cifre.

Notazione Scientifica

$r = \pm m \cdot \beta^e$ m = mantissa; e = 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}

Numeri Reali – Virgola mobile(2)

Rappresentazione di un numero binario in virgola mobile:

+1110.01 numero reale binario in virgola fissa

esponenti espressi in binario

$+1.11001 \cdot due^{+11}$ … … $+111001 \cdot due^{-10}$

(alcune possibili rappresentazioni in virgola mobile)

Numeri reali *normalizzati*:

- mantissa con parte intera costituita da un solo bit di valore 1
- rappresentazione è una tripla costituita da un bit e da due numeri naturali

Numeri Reali – Virgola mobile(3)

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

Standard IEEE 754-1985

s = codifica del segno (1 bit)
F = codifica della mantissa su M bit
E = codifica dell'esponente su K bit

$$r = (s == 0)?\ [+(1+f) \cdot due^{e})] : [-(1+f) \cdot due^{e})]$$

$f = F/2^M$ è la parte frazionaria della mantissa

$e = +E - (2^{K-1} - 1)$ è l'esponente rappresentato dal numero naturale E (*rappresentazione con polarizzazione*)

"uno implicito " $\Rightarrow$ lo zero non è rappresentabile

Numeri Reali – Virgola mobile(4)

Half precision a 16 bit con $K = 5$ e $M = 10$.

Esempio :
R = {1,10011,1110100101} rappresenta il numero reale negativo con:
$f = F/2^M = F/2^{dieci} = 0.1110100101$
$e = +E - (2^{K-1} - 1) = +10011 - (+01111) = +100$

$r = -1.1110100101 \cdot due^{+100} = -11110.100101$
r = -30.578125 in base *dieci*

Numeri Reali – Virgola mobile(5)

Esempio (numeri con modulo massimo, i cosiddetti $\pm\infty$):

R={*0*,11111,1111111111} (massimo numero rappresentabile)
R= {*1*,11111,1111111111} (minimo numero rappresentabile)

$f = F/2^M = F/2^{dieci} = 0.1111111111$
$e = +E - (2^{K-1} - 1) = +11111 - (+01111) = +10000$

$|r| = 1.1111111111 \cdot due^{+10000}$
$|r| = 1.31008 \cdot 10^{+5}$ in base *dieci*

Numeri Reali – Virgola mobile(6)

Esempio (numeri con modulo minimo, i cosiddetti ± 0):

R={*0*, 00000,0000000000} (minimo numero positivo)
R= {*1*,11111,1111111111} (massimo numero negativo)

$f = F/2^M = F/2^{dieci} = 0.0000000000$
$e = +E - (2^{K-1} - 1) = +00000 - (+01111) = -01111$

$|r| = 1.0000000000 \cdot due^{-01111}$
$|r| = 2^{-15} \cong 0.31 \cdot 10^{-4}$

Numeri Reali – Virgola mobile(7)

Standard IEEE 754-1985 prevede:

Rappresentazione in *single precision* su 32 bit con $K = 8$ e $M = 23$
Intervallo di rappresentabilita':

massimo modulo $6{,}8 \cdot 10^{+38}$
minimo modulo $2^{-127} \cong 0.58 \cdot 10^{-38}$

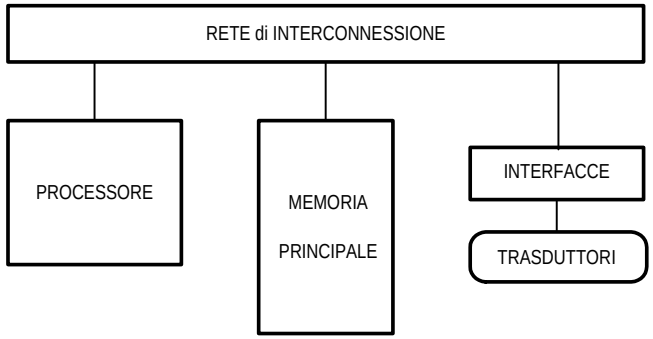
Rappresentazione in *double precision* su 64 bit con $K = 11$ e $M = 52$
Intervallo di rappresentabilita':

massimo modulo $+3.6 \cdot 10^{308}$
minimo modulo $2^{-1023} \cong 0.11 \cdot 10^{-307}$

IEEE 754-2008 aggiunge:
rappresentazione in *quadruple precision* su 128 bit con $K = 15$ e $M = 112$

Schema a blocchi di un semplice calcolatore

Architettura di von Neumann (1946)



Funzionamento

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

Il processore ripete all'infinito le azioni seguenti :

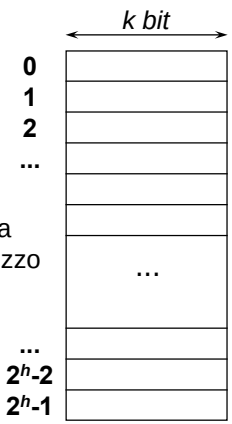
- preleva una nuova istruzione dalla memoria
- la decodifica
- la esegue

L'esecuzione di un'istruzione può comportare

» Elaborazione e/o Trasferimento (memoria ↔ processore, I/O ↔ processore)

Le periferiche permettono al calcolatore di interagire con il mondo esterno

Struttura logica della memoria

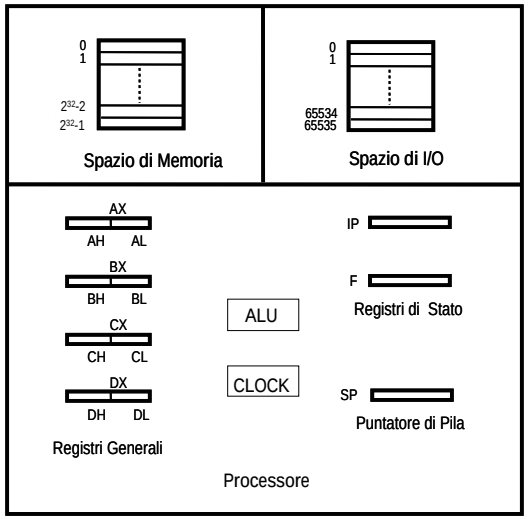


- 2^h *celle* di k bit ognuna
- Ogni cella ha un indirizzo

OPERAZIONI

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

Struttura logica del processore (1)



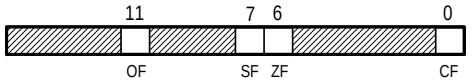
Struttura logica del processore (2)

Registro Stack Pointer SP

Registri di stato:

Instruction Pointer IP

Flag register F



Carry Flag CF

Zero Flag ZF

SignFlag SF

Overflow Flag OF

Formato delle Istruzioni:

ELAB destination, source

ELAB source

ELAB destination

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

Istruzioni di traslazione/rotazione

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

Istruzioni di controllo

Istruzioni di salto

JMP	Salto incondizionato
Jcond	Salto sotto condizione

Istruzioni per la gestione dei sottoprogrammi

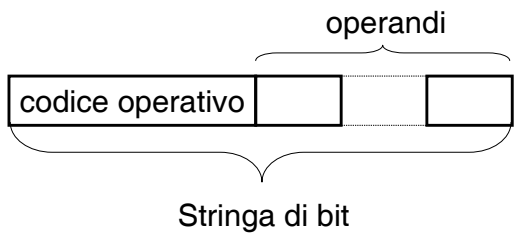
CALL	Chiamata di sottoprogramma
RET	Ritorno da sottoprogramma

Istruzione di alt

HLT	Alt

Formato delle istruzioni

Le istruzioni sono codificate come sequenze di bit.



Esempio

Codice Operativo **Operandi**

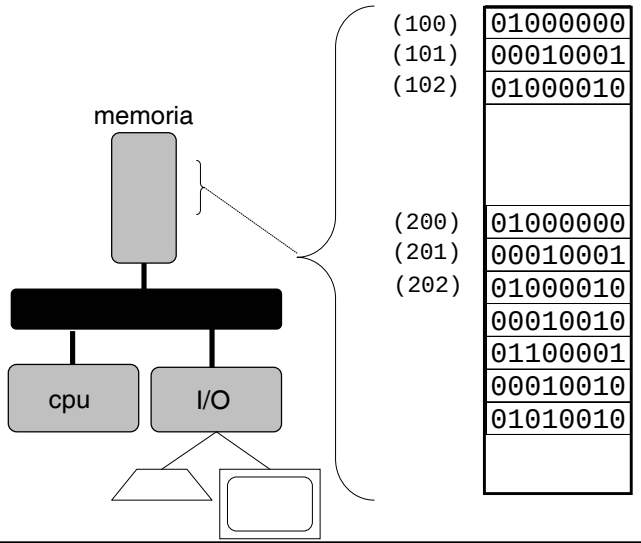
0000 **0001** **00110011**

MOV AL Cella 51

Esempio di programma (1)

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


Esempio di programma (2)



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 AL 0x54

Linguaggi ad Alto livello — indipendenti dalla macchina

```
int main()
{
    …
}
```


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;
}
```


Definizione di informatica

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

Informatica (definizione formale dell’Association for Computing Machine - 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

Calcolatore elettronico come risolutore di problemi

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 (1)

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.

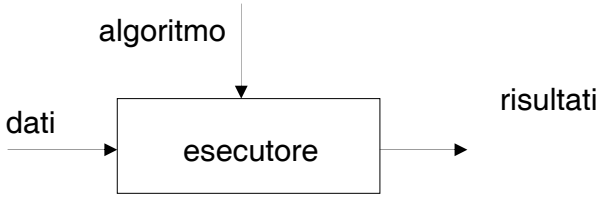
2. **Operazioni condizionali**
   Controllano una condizione. In base al valore della condizione, selezionano l'operazione successiva da eseguire.

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)

Algoritmo (3)

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*

Algoritmo (4)

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 corrispondeza degli stessi valori del dominio di ingressso 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

Algoritmo (6)

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 (6bis)

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: **MCD(m,n) = MCD((m-n),n)**

Algoritmo

- **ripeti finché** (*M != 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)

Algoritmo (7)

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**.**

Programmazione

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.

Linguaggi di Programmazione (2)

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 operazionale)
- mediante funzioni matematiche (semantica denotazionale)
- mediante formule logiche (semantica assiomatica)

Definizione di linguaggio

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 = <V, VN, P, S>

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 unica regola che indica l'**alternativa** fra A_1, …, A_n
…
A_n

Grammatica BNF (2)

G = <V, VN, P, S>

V = {lupo, canarino, bosco, cielo, mangia, vola, canta, ., il, lo}
VN = {***frase***, ***soggetto***, ***verbo***, ***articolo***, ***nome*** }
S = ***frase***

Produzioni P

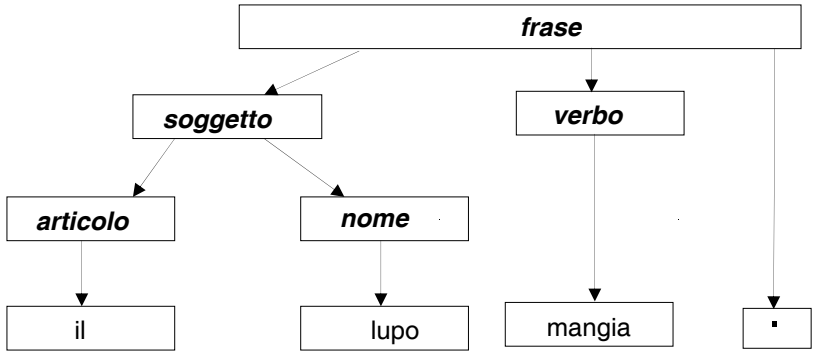
frase	::=	***soggetto verbo*** .
soggetto	::=	***articolo nome***
articolo	::=	il
		lo
nome	::=	lupo
		canarino
		bosco
		cielo
verbo	::=	mangia
		vola
		canta

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 << “immettrere due numeri” << endl;
   cin >> a;  cin >> b;
   int c = a + a;  ←
   cout << “Somma: “ << c << endl;
   return 0;
}
```


Approccio compilato

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…

Approccio Interpretato

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