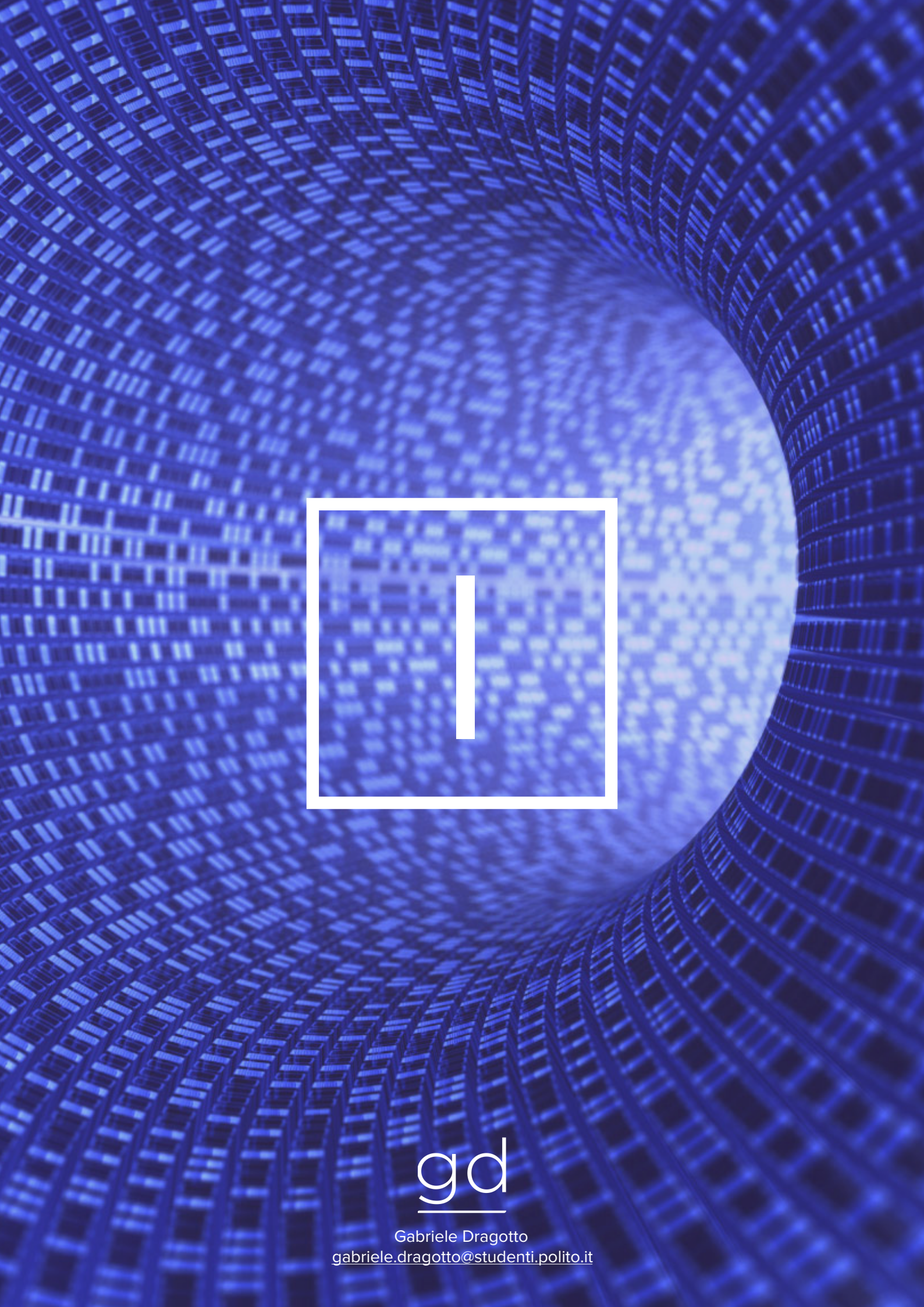


Gabriele Dragotto
gabriele.dragotto@studenti.polito.it

Introduzione

GENERAZIONI DI CALCOLATORI

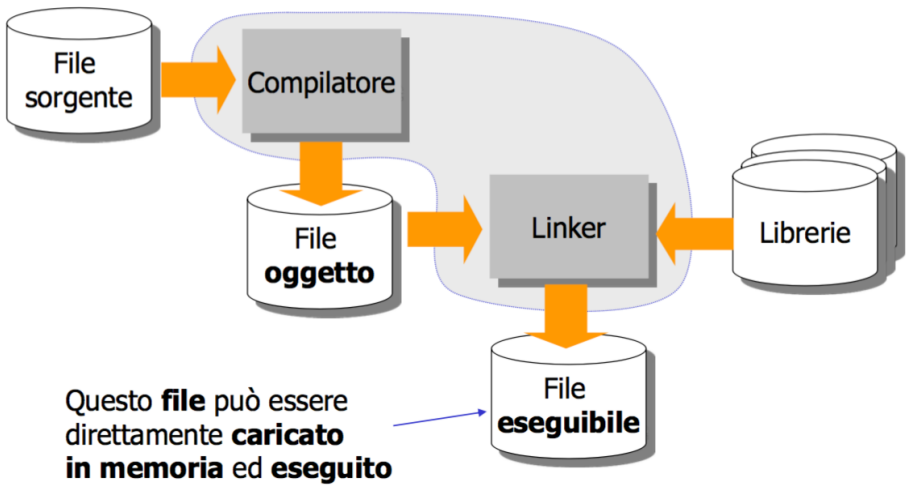
A partire dall'ENIAC, è possibile identificare alcune **generazioni** principali di computer, ciascuna facente riferimento a una diversa **tecnologia di base**

A. **TUBI A VUOTO**
   Le valvole termoioniche spesso si bruciano e non sono ottimali. Esse sono il primo componente attivo in un circuito
B. **TRANSISTOR**
   Nei BellLabs vengono progettati i primi transistor: due **terminali di input e uno di output.**
C. **MSI - CIRCUITI INTEGRATI**
   Elaborazione di schede integrate.
   Migliaia di transistors
D. **VLSI - CIRCUITI INTEGRATI**
   costo dell'esecuzione di 100K operazioni: **qualche milionesimo di dollaro**

ALGORITMO

Descrizione **formale di una sequenza finita di azioni** che devono essere eseguite per giungere alla soluzione di un problema

STRUTTURA PROGRAMMA



LIVELLO DI ASTRAZIONE

A. **ALTO LIVELLO**
   Elementi del linguaggio hanno complessità equivalente ai blocchi dei diagrammi di flusso strutturati
B. **BASSO LIVELLO**
   Microarchitettura (assembler)

Elementi del linguaggio

DEFINIZIONE

Essendo il linguaggio un'astrazione, esistono alcuni **fondamentali elementi sintattici** essenziali per l'uso del linguaggio stesso:

A. **KEYWORD**
Vocaboli "riservati" al traduttore per riconoscere altri elementi del linguaggio

B. **DATI**
Insieme di bit memorizzato in memoria centrale a cui **l'utente dà interpretazione.**
NOME + INTERPRETAZIONE + MODALITA_ACCESSO

C. **IDENTIFICATORI**
Indica il **nome di un dato** (e di altre entità) in un programma
Permette di dare nomi intuitivi ai dati

D. **TIPO**
Indica l'intepretazione dei dati in memoria

E. **ISTRUZIONI**
Indicano le **operazioni che il linguaggio permette di eseguire** (traducendole) a livello macchina:
PSEUDO_ISTRUZIONI + ISTRUZIONI ELEMENTATI + FLUSSO

SCOPE DELLE VARIABILI

Definisce la visibilità, globale o locale, della variabile dichiarata.
In C è possibile definire **a blocchi.**

DICHIARAZIONE

La dichiarazione di un dato richiede:
- **L'allocazione di uno spazio in memoria** atto a contenere il dato
- **L'assegnazione di un nome a tale spazio** in memoria

- Identificati da parole chiave!
  - `char` caratteri ASCII
  - `int` interi (complemento a 2)
  - `float` reali (floating point singola precisione)
  - `double` reali (floating point doppia precisione)

- La dimensione precisa di questi tipi dipende dall'architettura (non definita dal linguaggio)
  - `|char|` = 8 bit = 1 Byte sempre

`signed / unsigned` Dati di tipo char e int per il segno
`short / long` Dati di tipo int

- Interi
  - `[signed/unsigned] short [int]`
  - `[signed/unsigned] int`
  - `[signed/unsigned] long [int]`

- Reali
  - `float`
  - `double`

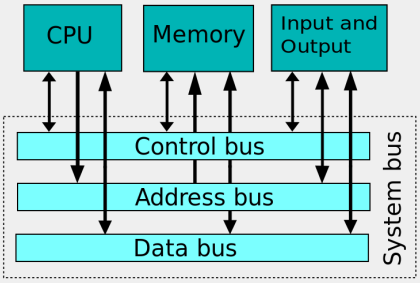
Architettura calcolatori

ARCH. VON NEUMAN

In informatica l'architettura di von Neumann è una tipologia di architettura hardware per computer digitali programmabili

DESCRIZIONE

1. **CPU**
2. **UNITA' DI MEMORIA**
3. **UNITA' DI INPUT**
4. **UNITA DI OUTPUT**
5. **BUS**



BUS

Unità di interconnessione e scambio dati
L'utilizzo di un bus favorisce la **modularità e l'espandibilità del calcolatore.**

ABUS
Address-BUS o bus degli indirizzi di memoria.

CBUS
Control-BUS o bus di controllo.

DBUS
Data-BUS o bus dei dati.

1. **MONODATO**
2. **FREQUENZA** = n. di dati trasportati al secondo
3. **AMPIEZZA** = n. di bit di cui è costituito un singolo dato

2^(|ABUS) * DBUS

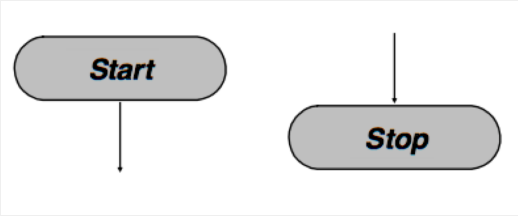
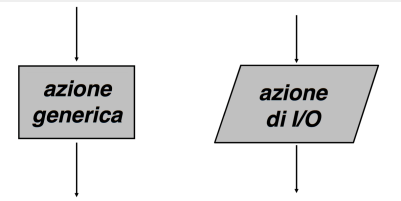
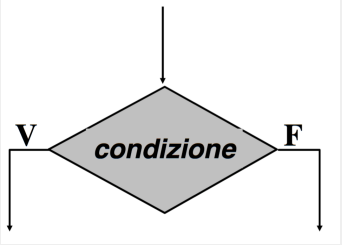
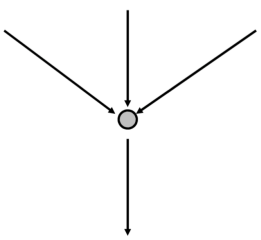
A. **GRADO DI PARALLELISMO**
Numero di bit della word
Coincide con la size del **DBUS**

B. **INDIRIZZABILITA'**
Numero di locazioni presenti nella memoria
2^(|ABus|)

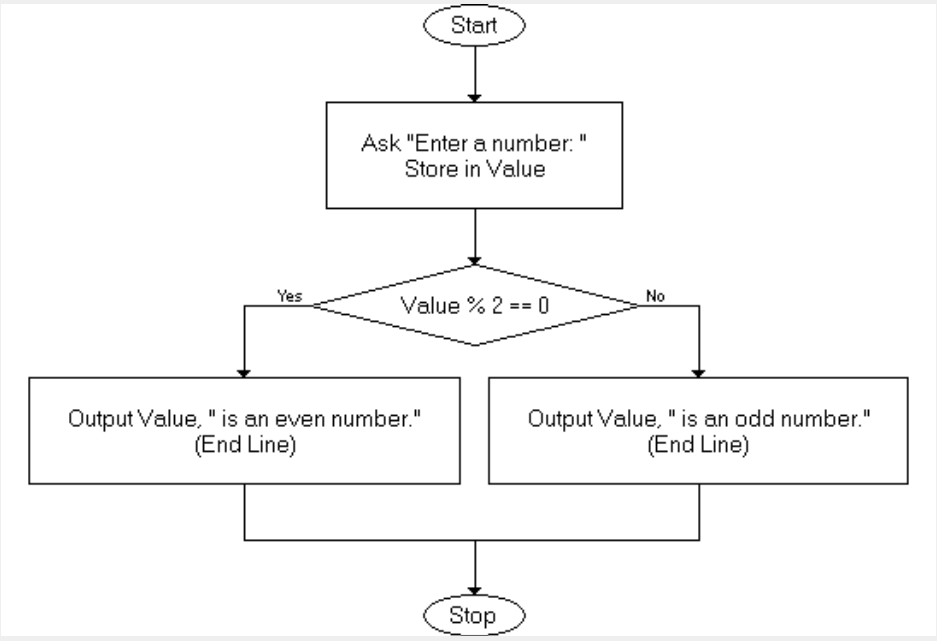
C. **CAPACITA'**
2^(|ABus|) * Parallelismo

CLOCK	Elemento centralizzato che crea un riferimento numerico per tutti i componenti interni. Ha **periodo e frequenza.**

Flow charts

DEFINIZIONE	Strumenti grafici che rappresentano **l'evoluzione logica** per la risoluzione di un problema dato.
COMPOSIZIONE	**A. BLOCCHI ELEMENTARI** Per descrivere azioni e decisioni. **B. ARCHI ORIENTATI** Per descrivere la sequenza di svolgimento delle azioni.
TEOREMA DI BOHM-JACOPINI	*"qualunque algoritmo può essere implementato in fase di programmazione* ***utilizzando tre sole strutture dette strutture di controllo****:* ***la sequenza, la selezione ed il ciclo****"*
TIPI DI BLOCCO	**BLOCCHI START/STOP**  **BLOCCHI DI AZIONE**  **BLOCCO DI SELEZIONE**  **BLOCCO DI CONNESSIONE** 

ODD OR EVEN



INPUT: READ X
OUTPUT: WRITE “THINGS”

OPERATORI

+ - * /

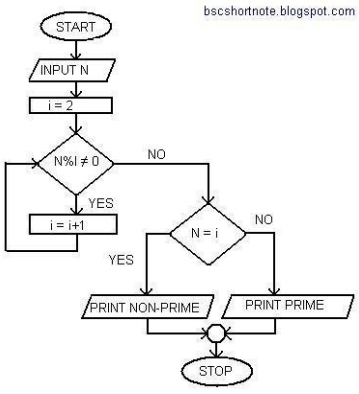
% —> **Modulo**, ovvero resto della divisione
31%2—> 1

ATTENZIONE
Nell’aritmetica intera non sono previste virgole.
31/2 = 15 — 31 % 2 = 1

STRUTTURE BASE

A. **UN SOLO START**
B. **UN SOLO STOP**
C. **SEQUENZA DI BLOCCHI**
D. **IF THEN ELSE**
E. **WHILE DO (Controllo in testa)**
F. **REPEAT UNTIL (Controllo in coda)**

PRIME NUMBER



Numeri

SISTEMA DECIMALE

252 = 2×100+5×10 +2×1
2x10^2 + 5x10^1 + 2x10^0

Si dice sistema **numerico posizionale.**

$$A = \sum_{i=0}^{N-1} a_j \cdot B^i$$

Non posizionali sono greco, **romano.**

FLIP FLOP

I **flip-flop** sono **circuiti elettronici sequenziali molto semplici**, utilizzati nell'elettronica digitale come dispositivi di memoria elementare. Il nome Flip-Flop deriva dal rumore che facevano i primi circuiti di questo tipo, costruiti con relè che permettevano il cambiamento di stato.

SISTEMA BINARIO

Alfabeto: {0,1}
I **transistor** si occupano di rilevare il micropassaggio positivo (1) o assente (0) di **corrente**.

101_2 =1×22 +0×21 +1×20 =1×4+1×1 = **5_{10}**

Ogni numero è contenuto in un **bit**. 8 bit compongono un pacchetto base, ovvero un **byte.**

13	**6**	**3**	**1**	**0**	*quozienti*
1	0	1	1		*resti*

$13_{10} = 1101_2$

252	0
126	0
63	1
31	1
15	1
7	1
3	1
1	1
0	0

$= 252_{10} = 11111100_2$

WORD

WORD
In informatica, word è un termine che identifica la **dimensione nativa dei dati usati da un computer**. Una word è semplicemente un gruppo di bit di una determinata dimensione che sono **gestiti come unità da un microprocessore**. La dimensione (o lunghezza) della word è un'importante caratteristica dell'architettura di un computer.

1Word (32bit) = 4B
1Word (64bit) = 8B

nibble (4 bit), byte (8 bit), dword o double word (32 bit), qword o quad word (64 bit).

ALCUNI TIPI DI DATO

1CHAR = 1B = 1B
1 WORD = 4B = 32BIT
1 FLOAT = 4B = 32BIT
1 DOUBLE = 8B = 64BIT

MSB E LSB

MSB Most Significant Bit, bit con peso maggiore
LSB Least Significant Bit, bit con peso minore

1 1 0 1 0 1

LIMITI SISTEMA BINARIO

$$\begin{cases} 0 \leq X \leq 2^{N-1} & BASE10 \\ (000...0) \leq X \leq (111...1) & BASE2 \end{cases}$$

1WORD = 32BIT = **4294967295**

SOMMA E SOTTRAZIONE

```
  1 1
  0 1 1 0 +
  0 1 1 1 =
-----------
  1 1 0 1
```

CARRY = BORROW = 1

```
  1  0  0  1  0  −
     1  1  0  1  =
-----------------
  0  0  1  0  1
```

Nella prima colonna ho 0-1 = 1 con **prestito di 1** da parte della seconda colonna

OVERFLOW UNDERFLOW

Si usa il termine **overflow** per indicare **l'errore** che si verifica in un sistema di calcolo automatico quando il **risultato di un'operazione non è rappresentabile** con la medesima codifica e numero di bit degli operandi.

1. **Numero di bit di allocazione** predefinito
2. **SOMMA** = **Carry** su MSB

```
 0101  +
 1110  =
 ----------
10011
```

overflow

SISTEMA ESADECIMALE

Alfabeto: {0,1,2...9,A,B,C,D,E,F}
Base: {16}
$CAFE_{(16)} = Cx16^3 + Ax16^2 + Fx16^1 + Ex16^0 = ...$

NUMERI SIGNED

A. MODULO E SEGNO

Il primo bit è significativo per il segno.

SEGNO + MODULO

1bit N -1 bit

$+3_{10} \rightarrow 0\ 011$

$-3_{10} \rightarrow 1\ 011$

B. COMPLEMENTO A DUE

Il complemento a due (in inglese **two's complement**) è il metodo più diffuso per la rappresentazione dei numeri con segno in informatica.

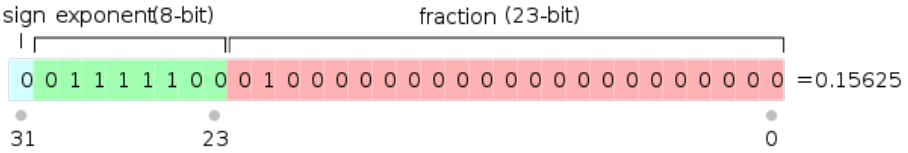
MSB = 1 => **Negativo**
MSB = 0 => **Positivo**
Il primo numero se è 1 pesa più della soma degli altri bit.

$101101 = 1\cdot(-2^5) + 0\cdot(2^4) + 1\cdot(2^3) + 1\cdot(2^2) + 1\cdot(2^0) = -19$

OVERFLOW: Segni numeri concordi e risultato con segno DISCORDE

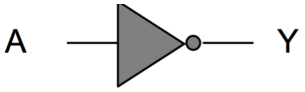
$$\begin{cases} N\_min = -\frac{B^i}{2} \\ N\_max = \frac{B^i}{2} - 1 \end{cases}$$

Per calcolare un numero negativo:
1.NUMERO IN BASE 2 MeS
2.SCAMBIO 0 CON 1
3. SOMMO 1

IEEE 754	*“Lo standard IEEE per il calcolo in virgola mobile è lo **standard** più diffuso nel campo del calcolo automatico.”*  **A. SEGNO** 1 Bit 0/1 **B. ESPONENTE** 8-11-15 Bit [-127,+126] **C. MANTISSA** 23-52-112 Bit Il numero può avere **precisione singola, doppia, quadrupla** in base al numero di bit della mantissa e esponente. L'esponente è pari a n, ma dobbiamo convertirlo in forma binaria e adattarlo allo standard. Per la precisione singola, dobbiamo aggiungere 127. Quindi 6 + 127 = 133. In forma binaria: 10000101.
CODIFICA ASCII	Usa 8 bit (originariamente 7 bit per US-ASCII) per rappresentare: - **52 caratteri alfabetici** (a...z A...Z) - **10 cifre** (0...9) - **segni di interpunzione** (,;!?...) - **Caratteri di controllo**

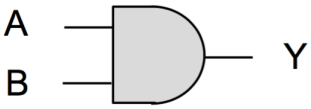
LOGICA BOOLEANA

NOT



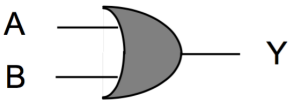
$Y = A'$

AND



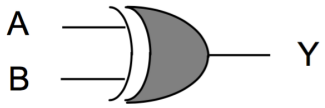
$Y = A \times B$

OR



$Y = A + B$

XOR



$Y = A \oplus B = A \times B' + A' \times B$

TEOREMA DI DE MORGAN

- $A + B = (A' \times B')'$
- $(A + B)' = A' \times B'$

Data types

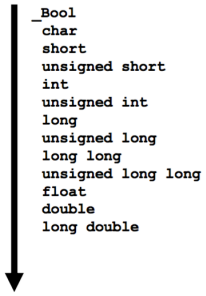
TABELLA I/O

Stringa di formato	Uso
%c	Variabili char
%d, %i	Valori in formato decimale
%x %X	Valori in formato esadecimale
%o	Valori in formato ottale
%l, %ld	Variabili long int
%u	Variabili unsigned
%f	Variabili float
%lf	Variabili double
%p	Indirizzo esadecimale di una variabile
%s	Stringhe di testo (le vedremo più avanti...)
%n	Scrive i byte scritti finora sullo stack dalla funzione printf() (molto sfruttata in contesti di format string overflow)

DATA TYPES

Tipo	Uso tipico	Dimensione (in bit) (riferimento: architettura x86)
char	Caratteri di testo ASCII, valori binari generici da 1 byte	8
short int	Numeri interi piccoli (da -32768 a 32767)	16
unsigned short int	Numeri positivi interi piccoli (da 0 a 65535)	16
int	Numeri interi (da -2147483648 a 2147483647)	32
unsigned int	Numeri interi positivi (da 0 a 4294967295)	32
long int	Numeri interi (la dimensione coincide con quella di un	32
	normale int su una macchina x86)	
long long int	Numeri interi grandi (da circa -9.22*10^18 a circa 9.22*10^18)	64
unsigned long long int	Numeri interi grandi positivi (da 0 a circa 1.84*10^19)	64
float	Numeri a virgola mobile (precisione singola)	32
double	Numeri a virgola mobile (doppia precisione, notazione scientifica)	64

Char: 1 byte = 8 bit;
Int: 2 byte = 16 bit (dipende dalla struttura dell'elaboratore);
Long: 4 byte = 32 bit (dipende dalla struttura dell'elaboratore);
Float: 4 byte = 32 bit;
Double: 8 byte = 64 bit (dipende dalla struttura dell'elaboratore);
Long double: 12 byte = 96 bit (dipende dalla struttura dell'elaboratore).

CAST	E’ possibile operare con variabili non dello stesso tipo promuovendole (cast) al livello più alto.  **(TYPE_CAST) variabile**

Prontuario funzioni

PRINTF	`int printf(char *format, arg list …)` Stampa l’argomento [0] sostituendo alle variabili i valori indirizzati in arglist.
SCANF	`int scanf(char *format, …)` Legge dallo stdin e **mette l'input negli indirizzi delle variabili** specificate nella lista di args; ritorna il numero di caratteri letti.

DO WHILE CODA	`#include <stdio.h>`
	`main() {`
	`int n; do`
	`  scanf (“%d“, &n);`
	`while (n <= 0);`
	`}`

COPIA VETTORI	`/* copia il contenuto di v[] in w[] */`
	`for( i=0; i<N; i++ )`
	`{`
	`w[i] = v[i] ; }`

FUNZIONI E PROCEDURE

```
int func1(int a);
<tipo risultato> <nome funzione> (<parametri formali >)
{ <istruzioni> }
```

`<tipo risultato> == void` se è una procedura.
`return` se è una procedura
`return value` se è una funzione

DICHIARARE IL PROTOTIPO FUNZIONE!

PARAMETRI FORMALI: Dichiarati nella definizione della func
PARAMETRI EFFETTIVI: Indirizzati nella chiamata alla func

PASSAGGI PER INDIRIZZO

E' possibile modificare lo schema di passaggio per valore in modo che i **parametri attuali vengano modificati** dalle istruzioni della funzione

FORMALI: Puntatori *
EFFETTIVI: Indirizzi **&**

ATTENZIONE:
Bisogna sempre specificare la **lunghezza del vettore** nelle funct.
I vettori **sono sempre passati per indirizzo, quindi modificati.**

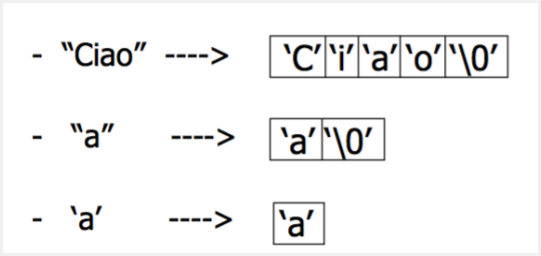
GETCHAR PUTCHAR

```
int getchar(void)
int putchar(char out)
```

Riceve l'argomento da stdin o stampa in stdout.

STRINGA

Sequenze di caratteri terminate dal carattere '\0' (NULL)
Vettori di caratteri terminati da un carattere aggiuntivo **'\0' (NULL)**



GETS PUTS

```
char *gets(*char s)
chat *puts(*char out)
```

Legge una riga da tastiera (fino al '\n')
La riga viene fornita come stringa (<stringa>), **senza il carattere '\n'**
In caso di errore, il risultato **è la costante NULL** (definita in stdio.h)

```
/* gets e scanf */
gets(nome)
scanf("%s",nome);
```

SSCANF SPRINTF

```
int main ()
{
  char sentence []="rudolph is 12 years old";
  char str [20];
  int i;

  sscanf (sentence,"%s %*s %d",str,&i);
  printf ("%s -> %d\n",str,i);

  return 0;
}
```

OPERAZIONI SU STRINGHE

funzione	***definizione***
`char* strcat (char* s1, char* s2);`	concatenazione s1+s2
`char* strchr (char* s, int c);`	ricerca di c in s
`int strcmp (char* s1, char* s2);`	confronto
`char* strcpy (char* s1, char* s2);`	s1 <= s2
`int strlen (char* s);`	lunghezza di s
`char* strncat (char* s1,char* s2,int n);`	concat. n car. max
`char* strncpy (char* s1,char* s2,int n);`	copia n car. max
`char* strncmp(char* dest,char* src,int n);`	cfr. n car. max

ARGOMENTI LINEA DI COMANDO

```
int main(int argc, char argv[] …)
```

```
main(int argc, char* argv[]) { int i, aflag=0, bflag=0;
char filename[80];
if (argc >= 2) { /* almeno due argomenti */
/* copiamo in una stringa, verra' aperto dopo */ strcpy
(filename, argv[argc-1]);
/* processiamo gli altri (eventuali argomenti) */ for
(i=1; i< argc-1; i++) {
          if (argv[i][0] == '-')  {/* e' un flag */
            switch (argv[i][1]) {
              case 'a':
                 aflag = 1; break;
case 'b':
bflag = 1; break;
default:
printf("Opzione non corretta.\n");
} }
```

STRUTTURE

```
struct identity {
char nome[30];
char cognome[30];
char codicefiscale[15]; int altezza;
   char statocivile;
}
```

Una struttura permette di accedere ai singoli campi tramite l'**operatore ' . ', applicato a variabil**i del corrispondente tipo `struct`

<variabile>.<campo>

E' POSSIBILE DEFINIRE UN TIPO

```
typedef struct complex {
double re;
double im;
} compl;

compl variabile = {0,0,0,null,...}
```

FOPEN

```
FILE* fopen(char* <nomefile>, char* <modo>);
```

- **<*modo*>**: Tipo di accesso al file
  - "r": sola lettura
  - "w": sola scrittura (cancella il file se esiste)
  - "a": *append* (aggiunge in coda ad un file)
  - "r+": lettura/scrittura su file esistente
  - "w+": lettura/scrittura su nuovo file
  - "a+": lettura/scrittura in coda o su nuovo file

- Ritorna:
  - Il puntatore al file in caso di successo
  - NULL in caso di errore

FUNZIONI VARIE

```
int getc (file* <file>);
int fgetc (FILE* <file>);
int putc (int c, file* <file>);
int fputc (int c, FILE* <file>);

char* fgets(char* <s>,int <n>,FILE* <file>);
```

A. Legge una stringa dal file f**ermandosi al più dopo n-1 caratteri**
B. L'eventuale **'\n' NON viene eliminato** (diverso da gets !)
C. Restituisce il puntatore alla stringa letta o NULL in caso di fine file o errore

LETTURA E SCRITTURA

```
int fscanf(FILE* <file>,char* <formato>,...);
int fprintf(FILE* <file>,char* <formato>,...);
void rewind (FILE* <file>)
```

- Esempio:

```
int a, *ptr;
ptr = &a;
```

Aritmetica dei puntatori (Cont.)

- Sottrazione tra puntatori:
  - Risultato corrisponde al numero di elementi del tipo cui si fa riferimento compresi tra i due puntatori
  - Esempio:

```
int *px, *py, diff;
/* assumiamo px=1000,  py = 1012*/
diff = px - py;
```

`Diff` non vale 12, bensì `(1012-1000)/sizeof(int)` **= 6**

Aritmetica dei puntatori

- Le operazioni sui puntatori non avvengono secondo l'aritmetica intera, ma dipendono dal tipo cui si fa riferimento

- Incremento/decremento di un puntatore:
  - Risultato ottenuto moltiplicando il valore dell'incremento o decremento per la dimensione del tipo cui si fa riferimento
  - Esempio:

```
int *px;
/* assumiamo px=1000 */
px += 3;
```

`px` non vale 1003, bensì `1000 + 3*sizeof(int)` = 1006

Puntatori e vettori: Analogie

- L' analogia tra puntatori e vettori si basa sul fatto che, dato un vettore `a[]`

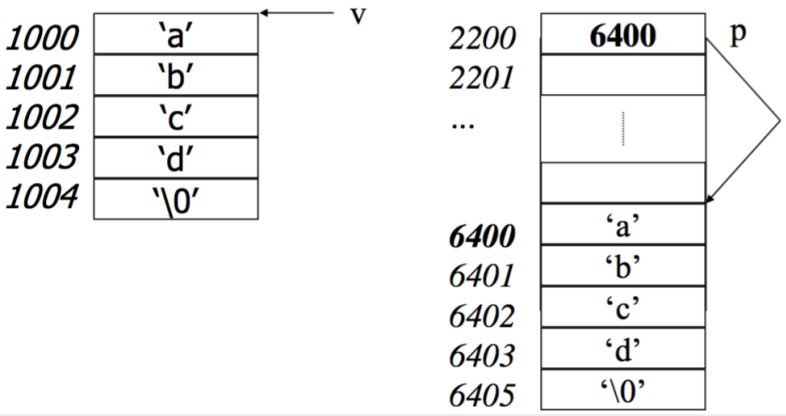
  `a[i]` e `*(aptr+i)` sono la stessa cosa

- Due modi per accedere al generico elemento del vettore:
  1. Usando l' array (tramite indice)
  2. Usando un puntatore (tramite scostamento o *offset*)

Interpretazione:

Puntatori e stringhe: Esempio

```
char v[] = "abcd";
char *p = "abcd";
```



Note finali

Alcuni dei contenuti presenti nelle seguenti dispense sono stati liberamente tratti dai materiali didattici disponibili al Politecnico di Torino.
Le dispense sono state elaborate dal sottoscritto come complemento allo studio e non intendono in alcun modo sostituire la completezza dei libri di testo e delle lezioni dalle quali sono state liberamente tratte.
Le dispense sono state scritte per l'esame di Informatica dell'A.A. 2015-2016, docente Paolo Bernardi.
E' doveroso quindi citare alcuni delle fonti da cui sono stati liberamente tratti alcune parti di esercizi e/o metodologie di soluzione:

- Paolo Bernardi, Whiteboard e appunti del corso di Informatica, A.A. 2015-2016.
- Marco Mezzalama, Whiteboard e appunti del corso di Informatica, A.A. 2015-2016.