Informatica - Mod. Programmazione
Lezione 10

Prof. Giuseppe Psaila

Laurea Triennale in Ingegneria Informatica
Università di Bergamo

Conversione da Base 10 a Base 2

- Si prende il numero in base 10
- Lo si divide per due:
  il resto è la cifra del numero binario (0 o 1)
  se il quoziente è zero, ci si ferma, altrimenti si procede con la divisione

$$
\begin{array}{r|ll}
13 & 1 & \quad 2^0 \\
6 & 0 & \quad 2^1 \\
3 & 1 & \quad 2^2 \\
1 & 1 & \quad 2^3 \\
0 & &
\end{array}
$$

Quindi, $13_{10} = 1101_2 = 8 + 4 + 1$

Esecizio

Convertire 22_{10} in base 2

Esercizio

Convertire 22_{10} in base 2

$$
\begin{array}{r|l}
22 & 0 \\
11 & 1 \\
5 & 1 \\
2 & 0 \\
1 & 1 \\
0 &
\end{array}
$$

Quindi, $22_{10} = 10110_2 = 16 + 4 + 2$

Base 8

- Ogni cifra ha un valore da 0 a 7
- Ogni cifra viene moltiplicata per una potenza crescente dell'8
- $217_8 = 2 \times 8^2 + 1 \times 8^1 + 7 \times 8^0 = 2 \times 64 + 8 + 7 =$
$$= 128 + 8 + 7 = 143_{10}$$

Da Base 10 a Base 8

- Come per la base 2, ma si divide per 8

143	7
17	1
2	2
0	

Da Base 10 a Base 8

Esercizio: Convertire 22_{10} in base 8

Da Base 10 a Base 8

Esercizio: Convertire 22_{10} in base 8

$$\begin{array}{r|l} 22 & 6 \\ 2 & 2 \\ 0 & \end{array}$$

$$26_8 = 2 \times 8 + 6 = 16 + 6 = 22_{10}$$

Base 16

- Ogni cifra ha un valore da 0 a 15 (16 valori), ma i valori sopra il 9 vengono indicati come $A = 10_{10}$, $B = 11_{10}$, $C = 12_{10}$, $D = 13_{10}$, $E = 14_{10}$, $F = 15_{10}$
- Ogni cifra viene moltiplicata per una potenza crescente del 16
- $8F_{16} = 8 \times 16^1 + 15 \times 16^0 = 128 + 15 = 143_{10}$

Da Base 10 a Base 16

- Come per la base 2, ma si divide per 16

143	F	(15)
8	8	
0		

$143_{10} = 8F_{16}$

Perché la Base 8?

Perché corrisponde a 3 cifre binarie

Base 2	Base 8
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Perché la Base 16?

Perché corrisponde a 4 cifre binarie

Base 2	Base 16	Base 2	Base 16
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Conversioni Veloci

- $8F_{16} = 1000\ 1111_2 = 10001111_2 = 010\ 001\ 111_2 = 217_8$
- $26_8 = 010\ 110_2 = 10110_2 = 0001\ 0110_2 = 16_{16}$

Somma bit a bit

$$\begin{array}{cc} 0 & + \\ 0 & = \\ \hline 0 & \end{array} \qquad \begin{array}{cc} 0 & + \\ 1 & = \\ \hline 1 & \end{array} \qquad \begin{array}{cc} 1 & + \\ 0 & = \\ \hline 1 & \end{array} \qquad \begin{array}{ccc} {}^{1} & 1 & + \\ & 1 & = \\ \hline & 0 & \end{array}$$

In base 10, $1 + 1 = 2$,ma 2 non è rappresentabile con un bit, bensì con due bit (10). Si innesca un **riporto** verso la cifra immediatamente a sinistra.

Somma bit a bit

$$
\begin{array}{cc} 0 & + \\ 0 & = \\ \hline 0 & \end{array}
\qquad
\begin{array}{cc} 0 & + \\ 1 & = \\ \hline 1 & \end{array}
\qquad
\begin{array}{cc} 1 & + \\ 0 & = \\ \hline 1 & \end{array}
\qquad
\begin{array}{ccc} {}^{1} & 1 & + \\ & 1 & = \\ \hline & 0 & \end{array}
$$

$$
\begin{array}{cc} 0^{1} & + \\ 0 & = \\ \hline 1 & \end{array}
\qquad
\begin{array}{ccc} {}^{1} & 0^{1} & + \\ & 1 & = \\ \hline & 0 & \end{array}
\qquad
\begin{array}{ccc} {}^{1} & 1^{1} & + \\ & 0 & = \\ \hline & 0 & \end{array}
\qquad
\begin{array}{ccc} {}^{1} & 1^{1} & + \\ & 1 & = \\ \hline & 1 & \end{array}
$$

Esercizio:

$$
\begin{array}{cccccc}
1 & 0 & 1 & 1 & 0 & + \\
0 & 0 & 0 & 1 & 1 & = \\
\hline
\end{array}
$$

Esercizio: (Soluzione)

$$
\begin{array}{cccccc}
1 & 0^1 & 1^1 & 1 & 0 & + \\
0 & 0 & 0 & 1 & 1 & = \\
\hline
1 & 1 & 0 & 0 & 1 &
\end{array}
$$

Esercizio:

$$
\begin{array}{ccccc}
0 & 1 & 1 & 1 & + \\
1 & 0 & 1 & 1 & = \\
\hline
\end{array}
$$

Operazioni Aritmetiche

Esercizio: (Soluzione)

$$
\begin{array}{cccccc}
^{1} & 0^{1} & 1^{1} & 1^{1} & 1 & + \\
 & 1 & 0 & 1 & 1 & = \\
\hline
1 & 0 & 0 & 1 & 0 &
\end{array}
$$

Operazioni Aritmetiche

Ma se i bit disponibili sono 4? $\Rightarrow$ Overflow

$$\begin{array}{c|ccccc} {}^{1} & 0^{1} & 1^{1} & 1^{1} & 1 & + \\ & 1 & 0 & 1 & 1 & = \\ \hline 1 & 0 & 0 & 1 & 0 & \end{array}$$

Il valore risultante NON È RAPPRESENTABILE con i bit disponibili

Limiti di Rappresentazione

- Numeri Naturali
- Con n bit,
  L'intervallo è $[0, 2^n - 1]$

Sottrazione bit a bit

$$\begin{array}{c} 0 \quad - \\ 0 \quad = \\ \hline 0 \end{array} \qquad \begin{array}{c} {}^{-1}\ 0 \quad - \\ 1 \quad = \\ \hline 1 \end{array} \qquad \begin{array}{c} 1 \quad - \\ 0 \quad = \\ \hline 1 \end{array} \qquad \begin{array}{c} 1 \quad - \\ 1 \quad = \\ \hline 0 \end{array}$$

$$\begin{array}{c} {}^{-1}\ 0\ {}^{-1} \quad - \\ 0 \quad = \\ \hline 1 \end{array} \qquad \begin{array}{c} {}^{-1}\ 0\ {}^{-1} \quad - \\ 1 \quad = \\ \hline 0 \end{array} \qquad \begin{array}{c} 1\ {}^{-1} \quad - \\ 0 \quad = \\ \hline 0 \end{array} \qquad \begin{array}{c} {}^{-1}\ 1\ {}^{-1} \quad - \\ 1 \quad = \\ \hline 1 \end{array}$$

Modulo e Segno

- Come rappresentare i numeri interi con segno?
- Non si possono introdurre due simboli in più (+ e -) come fa la matematica.
- Soluzione semplice: **Modulo e Segno**

Modulo e Segno

- Dati n bit, il bit più significativo (MSB, Most-Significant Bit) indica il segno:
  $0 =$ segno $+$
  $1 =$ segno -
- $n-1$ bit rimasti: **Valore Assoluto** (o **Modulo**)
- Intervallo di rappresentazione:
  $[-(2^{n-1}-1), -0][+0, 2^{n-1}-1]$

Modulo e Segno

- Esempio: su $n = 4$ bit
  $0100_{MS} = +4_{10}$
  $1100_{MS} = -4_{10}$
  $0000_{MS} = +0_{10}$
  $1000_{MS} = -0_{10}$
- Problemi:
  - doppia rappresentazione dello 0
  - complicato effettuare i calcoli

Complemento a Due

- Per evitare i problemi del sistema con Modulo e Segno si usa il sistema detto in **Complemento a Due**.
- È molto più efficace del Modulo e Segno sotto molti punti di vista
- Idea:
  - dati *n* bit, tutte le combinazioni con il primo bit a 0 rappresentano numeri positivi
  - tutte le combinazioni con il primo bit a 1 rappresentano numeri negativi
- Intervallo di rappresentazione: $[-2^{n-1}, +2^{n-1} - 1]$

Complemento a Due

- Si vuole fare in modo che il valore con il primo bit a 1 più grande rappresenti il valore negativo più grande: su $n = 4$ bit, $1111_{ca2} = -1_{10}$
- e il valore più piccolo rappresentabile con il primo bit a 1 rappresenti il valore negativo più piccolo rappresentabile:
  su $n = 4$ bit, $1000_{ca2} = -8_{10}$
- Nel mezzo, si mantiene l'ordine crescente dei numeri negativi

Complemento a Due

C. a 2	Base 10	C. a 2	Base 10
0111	+7	1111	-1
0110	+6	1110	-2
0101	+5	1101	-3
0100	+4	1100	-4
0011	+3	1011	-5
0010	+2	1010	-6
0001	+1	1001	-7
0000	0	1000	-8

Complemento a Due

- Legge matematica alla base:
  dato $A < 0$ da rappresentare,
  si calcola $B = 2^n - |A|$
  la conversione di B in binario è la rappresentazione di A in complemento a due
  Perché $-2^{n-1} \leq A \leq -1$, quindi $1 \leq |A| \leq 2^{n-1}$ e di conseguenza $(2^n - 2^{n-1}) \leq B \leq (2^n - 1)$, cioè $2^{n-1} \leq B \leq 2^n - 1$ (con $n = 4$ bit, $1000 \leq B \leq 1111$)

Complemento a Due

Esempi: $n = 4$ bit

- $A = -3_{10}$, $B = 16 - 3 = 13$, quindi $A = 1101_{ca2}$
- $A = -6_{10}$, $B = 16 - 6 = 10$, quindi $A = 1010_{ca2}$
- $A = -8_{10}$, $B = 16 - 8 = 8$, quindi $A = 1000_{ca2}$

Verificate sulla tabella (riportata nella slide successiva).

Complemento a Due

C. a 2	Base 10	C. a 2	Base 10
0111	+7	1111	-1
0110	+6	1110	-2
0101	+5	1101	-3
0100	+4	1100	-4
0011	+3	1011	-5
0010	+2	1010	-6
0001	+1	1001	-7
0000	0	1000	-8

Operazione di Inversione di Segno

Dato un numero A, otteniamo $-A$:

- Da A, si deriva $\overline{A}$ complementando (invertendo) ogni bit
- Quindi, $-A = \overline{A} + 1$

Esempi

- $A = 0011_{ca2} = +3_{10}$, $\overline{A} = 1100$, $-A = 1100 + 1 = 1101_{ca2}$
- $A = 1001_{ca2} = -7_{10}$, $\overline{A} = 0110$, quindi $-A = 0110 + 1 = 0111_{ca2}$
- $A = 0010_{ca2} = +2_{10}$, $\overline{A} = 1101$, quindi $-A = 1101 + 1 = 1110_{ca2}$

Verificate sulla tabella (riportata nella slide successiva).

Complemento a Due

C. a 2	Base 10	C. a 2	Base 10
0111	+7	1111	-1
0110	+6	1110	-2
0101	+5	1101	-3
0100	+4	1100	-4
0011	+3	1011	-5
0010	+2	1010	-6
0001	+1	1001	-7
0000	0	1000	-8

Operazione di Inversione di Segno Veloce

Dato un numero A, otteniamo $-A$:

- Si cerca la posizione p da destra che contiene il primo 1 (da destra)
- Ogni bit in posizione $i \leq p$ (da destra) rimane uguale in $-A$,
  gli altri si invertono

Esempi

- $A = 0011_{ca2} = +3_{10}$, $001\underline{1}$, quindi $-A = 1101_{ca2}$
- $A = 1001_{ca2} = -7_{10}$, $100\underline{1}$, quindi $-A = 0111_{ca2}$
- $A = 0010_{ca2} = +2_{10}$, $00\underline{1}0$, quindi $-A = 1110_{ca2}$
- $A = 1100_{ca2} = -4_{10}$, $1\underline{1}00$, quindi $-A = 0100$
- $A = 1000_{ca2} = -8_{10}$, $\underline{1}000$, quindi $-A = 1000_{ca2}$ (perché $+8$ non è rappresentabile)

Operazioni di Somma e Differenza

- $Z = A + B$
  si ignora l'overflow
  se A e B sono discordi, la somma è certamente rappresentabile
  se A e B sono concordi e Z è discorde, la somma non è rappresentabile
- La differenza si fa nel seguente modo:
  $Z = A - B = A + (-B)$

Complemento a Due

Esempio

$Z = +7_{10} + (-5_{10})$

$$\begin{array}{c|ccccc} ^{1} & 0^1 & 1^1 & 1^1 & 1 & + \\ & 1 & 0 & 1 & 1 & = \\ \hline & 0 & 0 & 1 & 0 & \end{array}$$

La somma di due numeri discordi dà sempre un valore rappresentabile

Esempio

$Z = +7_{10} + (+1_{10})$

$$
\begin{array}{c|ccccc}
 & 0^1 & 1^1 & 1^1 & 1 & + \\
 & 0 & 0 & 0 & 1 & = \\
\hline
 & 1 & 0 & 0 & 0 &
\end{array}
$$

La somma non è rappresentabile, ma si ottiene, come effetto, il numero più piccolo rappresentabile (rimbalzo)

Esempio

$Z = -8_{10} - (+1_{10}) = -8_{10} + (-1_{10})$

$$
\begin{array}{c|ccccc}
{}^{1} & 1 & 0 & 0 & 0 & + \\
 & 1 & 1 & 1 & 1 & = \\
\hline
 & 0 & 1 & 1 & 1 &
\end{array}
$$

La somma non è rappresentabile, ma si ottiene, come effetto, il numero più grande rappresentabile (rimbalzo)

Domanda da Tema d'Esame (1/2)
Si calcoli il valore di Z in complemento a 2 su 8 bit e lo si converta in Base 10.

$Z = X - Y$, con $X = BA_{16}$ e $Y = EE_{16}$
cioè $Z = BA_{16} - EE_{16} = 10111010_{ca2} - 11101110_{ca2} =$
$= 10111010_{ca2} + 00010010_{ca2}$

$$\begin{array}{c|cccccccc c}
 & 1 & 0^1 & 1^1 & 1 & 1 & 0^1 & 1 & 0 & + \\
 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & = \\
\hline
 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 &
\end{array}$$

$Z = 11001100_{ca2}$

Domanda da Tema d'Esame (2/2)

Ma $Z = 11001100_{ca2}$ è negativo,
allora calcoliamo $|Z| = -Z = 00110100_{ca2} =$
$= 32 + 16 + 4 = 52_{10}$

Risulta quindi $Z = 11001100_{ca2} = -52_{10}$

Numeri con Parte Frazionaria

- Se vogliamo rappresentare la parte frazionaria dei numeri, usiamo le potenze negative del 2
- $A = 110.101 = 6 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} =$
  $6 + 0,5 + 0,125 =$
  $= 6,625_{10}$
- Conversione della parte frazionaria dalla Base 10 alla Base2:
  - si moltiplica per 2,
  - la parte intera è la cifra binaria,
  - la parte frazionaria viene moltiplicata per 2, finchè non si ottiene 0 (se si ottiene)

Da Base 10 a Base 2

$6,625_{10}$

$$\begin{array}{r|l} 0{,}625 & 1 \\ 0{,}25 & 0 \\ 0{,}5 & 1 \\ 0 & \end{array}$$

$A = 110.101_2$

Da Base 10 a Base 2

$5,375_{10}$

$$\begin{array}{r|l} 0{,}375 & 0 \\ 0{,}75 & 1 \\ 0{,}5 & 1 \\ 0 & \end{array}$$

$A = 101.011_2$

Da Base 10 a Base 2

$5,374_{10}$

$$\begin{array}{r|l} 0{,}374 & 0 \\ 0{,}748 & 1 \\ 0{,}496 & 0 \\ 0{,}992 & 1 \\ 0{,}984 & \\ \ldots & \end{array}$$

$A \simeq 101.0101_2$

Principio

- Si basano sulla notazione esponenziale
- $A = 101.011 =$
  $= 101.011 \times 2^0 = 10.1011 \times 2^1 = 0.101011 \times 2^3 =$
  $= 1010.11 \times 2^{-1} = 101011 \times 2^{-3}$
- Forma generale: $m \times 2^e$
  m è detto **Mantissa**
  e è l'**Esponente**

Rappresentazione

- Si usano 32 bit
- Un bit per il segno s, che vale 0 o 1 a seconda che il segno del numero sia $+$ o -
- 8 bit per E, che rappresenta l'esponente modulo 127 (quindi, $E = e + 127$)
- 23 bit per rappresentare la mantissa, dove $m = 1.M$ (la potenza è allineata per avere il numero come $1.xxx \times 2^e$)

Esempio: 110.011×2^0

- Si allinea l'esponente:
  $110.011 \times 2^0 = 1.10011 \times 2^2$
  $m = 1.10011$
  $e = 2$
- $E = e + 127 = 129 = 10000001_2$
- $M = 10011000000000000000000$ (cifre a destra del punto in $m = 1.10011$

0	10000001	10011000000000000000000
s	E	M

Quindi $A = 01000000110011000000000000000000$

Oltre la Tabella ASCII

- Consorzio UNICODE
  Ha codificato i caratteri delle lingue mondiali ufficiali
- Per far questo, i codici dei caratteri superano il limite superiore di 127 della tabella ASCII
- Sito ufficiale
  `https://home.unicode.org/`
- Sezione tecnica
  `http://unicode.org/main.html`

Problema dell'Encoding

- Come scrivere documenti che usano codici di caratteri superiori a 127?
- Si potrebbe pensare di usare sempre 4 Byte per ogni carattere, ma così si sprecherebbe spazio
  inoltre non si avrebbe la retro-compatibilità
- **Encoding dei caratteri**: si usa un formato che incapsula il codice,
  rendendo chiaro quanti Byte devono essere letti.
- **UTF-8**
  Unicode Transformation Format 8 bit

UTF-8

- 4 configurazioni, a 1, 2, 3 e 4 Byte.
- I prefissi dei Byte sono impostati in modo da indicare chiaramente:
  - Quanti sono i Byte della sequenza
  - Se il Byte in questione è il primo Byte della sequenza oppure no

UTF-8 - Configurazione a 1 Byte

- Struttura: 0*xxxxxxx* (*x* indica un valore generico del bit)
- Con 7 bit utili, corrisponde ai caratteri della Tabella ASCII
- Codici fino a $2^7 = 127$.

UTF-8 - Configurazione a 2 Byte

- Struttura: 110*xxxxx* 10*xxxxxx*
- Con 11 (5+6) bit utili,
- Codici da 128 a $2^{11} - 1 = 2047$.

UTF-8 - Configurazione a 3 Byte

- Struttura: 1110*xxxx* 10*xxxxxx* 10*xxxxxx*
- Con 16 (4+6+6) bit utili,
- Codici da 2^{11} a $2^{16} - 1 = 65535$.

UTF-8 - Configurazione a 4 Byte

- Struttura: 11110*xxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
- Con 21 (3+6+6+6) bit utili,
- Codici da 2^{16} a $2^{21} - 1 = 2.097.151$.

UTF-8 - Come fare l'Encoding

- Si prende il codice del carattere
- Si identifica la configurazione minima necessaria
- Si allinea il codice ai bit utili della configurazione (aggiungendo 0 a sinistra)
- Si ripartiscono i bit nei Byte della configurazione

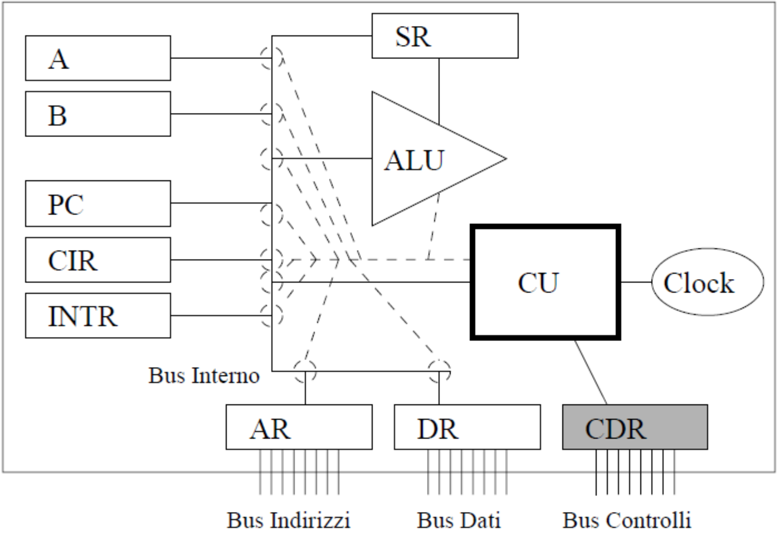
UTF-8 - Come fare l'Encoding

- Esempio: Carattere $c = 130_{10} = 10000010_2$ (8 bit)
- Usiamo la configurazione a 2 Byte (fino a 11 bit utili)
- Allineiamo c a 11 bit:
  $c = 000\ 10000010_2 = 00010\ 000010$
- Facciamo l'Encoding
  11000010 10000010
  cioè $110\overline{00010}\ 10\overline{000010}$

UTF-8 - Come fare il Decoding

- Esempio: $e = 11100000\ 10100000\ 10000000$ (configurazione a 3 Byte)
- Estraiamo i bit utili: da
  $e = 1110\overline{0000}\ 10\overline{100000}\ 10\overline{000000}$
  cioè $c = 0000\ 100000\ 000000$
- Eliminiamo i bit in eccesso:
  $c = 100000000000_2 = 2^{11} = 2048_{10}$

Architettura della CPU



Architettura della CPU

- CU: Control Unit
- PC: Program Counter, indirizzo della prossima istruzione da eseguire
- CIR: Current Instruction Registry, contiene l'istruzione in esecuzione
- A e B: registri dati
- SR: State Registry, i vari bit indicano che cosa è successo nell'ultima istruzione eseguita

Architettura della CPU

- ALU: Artihmetic-Logic Unit, effettua i calcoli matematici e i confronti
- Clock: Orologio di sistema, genera un segnale a Onda Quadra a frequenza costante, per cadenzare i tempi di apertura e chiusura dei collegamenti sul Bus Interno
- INTR: Interrupt Registry, registro interruzione, riporta il codice dell'interruzione da gestire

Interfaccia con il Bus di Sistema

- AR: Adress Registry, registro indirizzi, contiene l'indirizzo della cella di memoria cui accedere
- DR: Data Registry, registro dati, contiene il dato da inviare o ricevuto dalla cella di memoria
- CDR: Control Driver, viene pilotato dalla CU per inviare il segnale di controllo che indica l'operazione da svolgere (es. READ o WRITE)

CU e Micro-Istruzioni

- L'unità di controllo carica un'istruzione dalia memoria, la riconosce e la esegue.
- Tutte queste attività sono descritte da una sorta di "Programma Interno' detto **Micro-Programma**, fatto da **Micr-Istruzioni**
- Le Micro-Istruzioni eseguono operazioni di bassissimo livello, grazie alle quali vengono eseguite le fasi di acquisizione ed esecuzione delle istruzioni

Esempio: La Fase di Fetch delle Istruzioni

- Questa fase precede la vera esecuzione di ogni istruzione, perchè acquisisce l'istruzione da eseguire dalla memoria centrale
- Micro-Programma:
  PC $\Rightarrow$ AR
  READ
  DR $\Rightarrow$ CIR
  PC $+ 1 \Rightarrow$ PC

Esempio di istruzioni in Linguaggio Macchina

- LOADA indirizzo
  LOADB indirizzo
- STOREA indirizzo
  STOREB indirizzo
- SUM
  INC
- JUMP indirizzo
  JUMPZ indirizzo
  JUMPGZ indirizzo

Esempio di istruzioni in Linguaggio Macchina

- Istruzione in C++

```
x += y;
```

- In linguaggio macchina:

```
LOADA ind_x
LOADB ind_y
SUM
STOREA ind_x
```