Capitolo 6

Notazione posizionale

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

- Ci sono solo 10 tipi di persone al mondo:
  - quelle che conoscono la rappresentazione dei numeri in base 2, e
  - quelle che non la conoscono ...

Base 2

- Per capire fino in fondo come sono rappresentate le informazioni in un calcolatore occorre conoscere la rappresentazione dei numeri in base 2
- Il motivo è che le informazioni sono rappresentate come sequenze di bit, ossia cifre con due soli possibili valori

Basi e cifre 1/2

- Partiamo dalla rappresentazione di un numero in una generica base
- Cominciamo dalla rappresentazione dei numeri naturali

Basi e cifre 2/2

- Rappresentazione di un numero in una data base: sequenza di cifre
- *Cifra*: simbolo rappresentante un numero
- *Base*: numero (naturale) di valori possibili per ciascuna cifra
- In base $b > 0$ si utilizzano b cifre distinte, per rappresentare i valori $0,\ 1,\ 1+1,\ 1+1+1,\ \ldots,\ b-1$

Cifre e numeri in base 10

- Es: in base 10 le cifre sono

0 che rappresenta il valore 0

1 che rappresenta il valore 1

2 che rappresenta il valore 1+1

3 che rappresenta il valore 1+1+1

Simbolo grafico

Concetto astratto di numero naturale

.

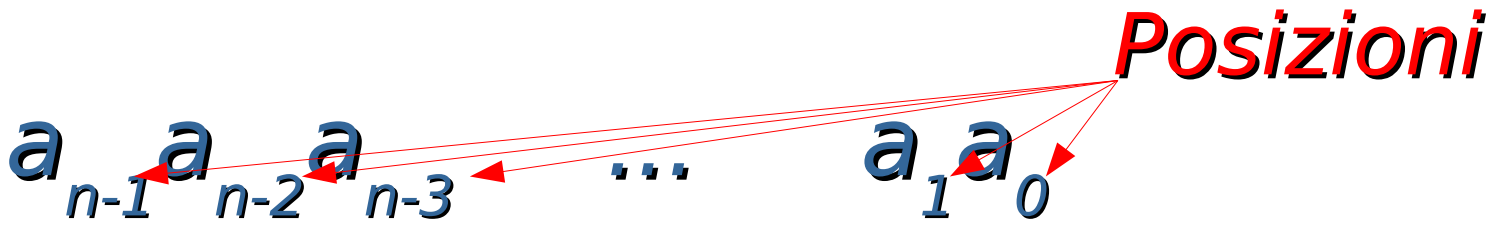
.

.

9 che rappresenta il valore 1+1+1+1+1+1+1+1+1

Notazione posizionale

- Rappresentazione di un numero su *n* cifre in base *b*:



$$a_i \in \{0, 1, \ldots, b - 1\}$$

- Es: Notazione decimale:

$b = 10, a_i \in \{0, 1, 2, \ldots, 9\}$

$345 \Rightarrow a_2 = 3, a_1 = 4, a_0 = 5$

Notazione

- Per rendere esplicita la base utilizzata, si può utilizzare la notazione

$$[\, x \,]_b$$

$$a_i \in \{0, 1, \ldots, b - 1\}$$

dove *x* è una qualsiasi espressione, ed il cui significato è che ogni numero presente nell'espressione è rappresentato in base *b*

Esempi in base 10

$[\ 345\ ]_{10}$

$[\ 2 * 10 + 5 * 1\ ]_{10}$

Notazione posizionale

$$[\ a_{n-1}a_{n-2}a_{n-3}\ldots a_1a_0\ ]_b =$$

$$[a_0*1 + a_1*b + a_2*b^2 + a_3*b^3 + \ldots + a_{n-1}*b^{n-1}\ ]_b$$

$$= [\ \sum_{i=0,\,1,\,\ldots,\,n-1} a_i * b^i\ ]_b$$

Peso cifra i-esima

- Es: $b = 10$, $a_i \in \{0, 1, 2, \ldots, 9\}$

$$[\ 345\ ]_{10} = [3 * 10^2 + 4 * 10 + 5 * 1]_{10}$$

"yo cuento como un cero a la izquierda"
... io conto come uno zero a sinistra

Calcoli

- Si utilizzano degli algoritmi
- Esattamente quelli imparati alle elementari per la base 10
- Esempio: per sommare due numeri, si sommano le cifre a partire da destra e si utilizza il riporto

Notazione binaria

- Base 2, 2 cifre:
  - 0, 1
- La cifra nella posizione *i-esima* ha peso 2^i
- Esempi (*configurazioni di bit*):

$[0]_{10} = [0]_2$

$[1]_{10} = [1]_2$

$[2]_{10} = [10]_2 = [1*2 + 0*1]_{10}$

$[3]_{10} = [11]_2 = [1*2 + 1*1]_{10}$

Video

- https://www.facebook.com/watch/?v=2484427684944430

Base 16

- Una base che risulta spesso molto conveniente è la base 16
- Vediamo prima di cosa si tratta, e poi come mai è molto utilizzata

Notazione esadecimale

- Base 16, 16 cifre:
  - 0, 1, 2, …, 9, A, B, C, D, E, F
- Valore cifre in decimale:
  - 0, 1, 2, …, 9, 10, 11, 12, 13, 14, 15
- La cifra nella posizione *i-esima* ha peso 16^i
- Esempi:

$$[0]_{10} = [0]_{16}$$

$$[10]_{10} = [A]_{16}$$

$$[18]_{10} = [12]_{16} = [1*16 + 2*1]_{10}$$

Motivazione Base 16 1/3

- Ogni cifra in base sedici corrisponde ad una delle possibili combinazioni di 4 cifre in base 2
- Quindi, data la rappresentazione in base 2 di un numero naturale, la sua rappre-sentazione in base 16 si ottiene dividendo la sequenza in base in sotto-sequenze consecutive da 4 cifre ciascuna, partendo da destra, e convertendo ciascuna sotto-sequenza di quattro cifre binarie nella corrispondente cifra in base 16

Motivazione Base 16 2/3

Esempio: Dato il numero
$[1000001111]_2$

Dividiamo le cifre in gruppi da quattro da destra:
10 0000 1111
ed aggiungiamo due zeri all'inizio (senza modificare il valore del numero):
0010 0000 1111

In base 16 otteniamo:
2 0 F

Motivazione Base 16 3/3

- Viceversa, data la rappresentazione in base 16 di un numero naturale, il corrispondente numero in base 2 si ottiene convertendo semplicemente ciascuna cifra della rappresentazione in base 16 nella corrispondente sequenza di 4 cifre in base 2
- Invertendo il precedente esempio: $[20F]_{16} = [1000001111]_2$

Rappresentazione naturali

- In una cella di memoria o in una sequenza di celle di memoria si può memorizzare con facilità un numero naturale memorizzando la configurazione di bit corrispondente alla sua rappresentazione in base 2
- Questa è la tipica modalità con cui sono memorizzati i numeri naturali
- Coincide con gli esempi che abbiamo già visto in lezioni precedenti

Conversioni di base

- Come abbiamo visto, un numero (entità astratta) e la sua rappresentazione (sequenza di cifre scritta concretamente da qualche parte) sono due entità distinte
- Nelle slide precedenti abbiamo visto esempi di conversioni tra rappresentazioni
- Affrontiamo ora l'argomento in senso generale
- Per il caso più semplice: numeri naturali

Conversione numero/rappres.

- Il passaggio dalla rappresentazione in una base alla rappresentazione in un'altra base può essere realizzato utilizzando uno dei seguenti algoritmi di conversione, o combinandoli in sequenza, in base all'informazione iniziale di cui si dispone
  - Da un numero naturale *N* alla sua rappresentazione in una qualche base
  - Dalla rappresentazione di un numero naturale *N* in una qualche base al numero *N*

Assunzione

- Per entrambe le conversioni, assumiamo che il numero naturale *N* sia memorizzato in un elaboratore in grado di effettuare i calcoli richiesti dall'algoritmo
- Tale numero è quindi a sua volta rappresentato in qualche modo nella memoria dell'elaboratore
  - Ma non ci interessiamo di tale rappresentazione (astraiamo)
  - Assumiamo solo che l'elaboratore sia in grado di rappresentare tale numero, e di fare calcoli su tale numero

Esempio

- Un esempio è un numero naturale in un tipico elaboratore
- Il numero è rappresentato in base 2
- L'elaboratore è in grado di fare calcoli con i numeri naturali
  - Noi sfruttiamo questa proprietà per scrivere programmi con numeri naturali, senza aver bisogno di interessarci di come sono rappresentati esattamente tali numeri

Da numero a rappresentazione

- Data un qualsiasi base *b*, e la sua rappresentazione, come alla slide 7
- Il valore della cifra in posizione *i*-esima è data dalla seguente formula (di facile verifica):

  $(N / b^i) \% b$

- I calcoli nella formula sono effettuabili dall'elaboratore, come abbiamo assunto

Da valore a simbolo cifra 1/2

- Ovviamente la precedente formula ci da il valore della cifra, che a sua volta è un numero
  - Tale numero è memorizzato nell'elaboratore, in una qualche base
- A noi però interessa la cifra, ossia il simbolo relativo a quel valore
- Per ottenere tale simbolo potrebbe si utilizzare ad esempio una tabella, che fa corrispondere il carattere giusto a ciascun valore

Da valore a simbolo cifra 2/2

- Ad esempio, se le base *b* è 16, possiamo utilizzare la seguente tabella

$[0]_{10}$	0
$[1]_{10}$	1
$[2]_{10}$	2
...	
$[9]_{10}$	9
$[10]_{10}$	A
$[11]_{10}$	B
...	
$[15]_{10}$	F

Esempio

- Quanto detto finora può essere utilizzato, ad esempio, per convertire facilmente un numero in base 10 in un numero in un'altra base, nel caso del linguaggio C/C++
- Infatti, sia il compilatore che le funzioni di libreria per l'ingresso si aspettano che i numeri siano scritti in base 10
  - Tali numeri sono poi rappresentati tipicamente in base 2 nell'elaboratore, ma non ci interessiamo di questo dettaglio
- Quindi, una volta inserito o letto un numero in base 10 nel programma o nel processo, basta utilizzare la precedente formula per ottenere le cifre di tale numero in una qualsiasi base

Esercizio

- Trovate la traccia completa di un esercizio di conversione da base 10 a base 2 nel file *base2.txt*, tra i compiti per casa della esercitazione 5

Da rappresentazione a numero

- Questa volta si dispone solo della rappresentazione, ossia della sequenza di cifre (simboli) in una qualche base *b*
- Con una tabella inversa rispetto a quella che abbiamo utilizzato per ottenere le cifre (caratteri) dai numeri, traduciamo ciascuna cifra nel corrispondente numero naturale
- L'elaboratore può fare calcoli su tali numeri naturali (che sono rappresentati in qualche modo dentro l'elaboratore)
- Basta quindi sostituire tali valori delle cifre nella formula alla slide 10

Esempio

- Questo algoritmo può essere utile, ad esempio, per calcolare i numeri corrispondenti a rappresentazioni scritte in basi numeriche che non ci è consentito utilizzare nei programmi in C/C++
- Una tale base è la base 2
- Quindi possiamo utilizzare questo algoritmo per convertire
  - da rappresentazioni in base 2
  - a numeri memorizzabili nei nostri programmi
- Trovate un esempio di tale conversione tra le tracce in *base2.txt*

Rappresentazione interi 1/2

- Come rappresentare però numeri con segno?
- Non esiste un elemento all'interno delle celle, che sia destinato a memorizzare il segno
- Come potremmo cavarcela?

Rappresentazione interi 2/2

- Un'idea sarebbe quella di utilizzare uno dei bit per il segno
  - 0 per i valori positivi
  - 1 per i valori negativi
- Il problema è che sprechiamo una configurazione di bit, perché avremmo due diverse rappresentazioni per il numero 0
  - Una col segno positivo
  - Una col segno negativo

Idea

- Rappresentare i numeri positivi semplicemente in base 2
- Non rappresentare i numeri negativi direttamente, ma sommargli prima una costante, che fa si che diventino positivi
- Il trucco starà nel far sì che i **veri** numeri positivi cadano in un intervallo di valori diverso da quello in cui cadono i **falsi** numeri positivi (ossia quelli ottenuti sommando una costante)
- Questa idea è alla base della **rappresentazione in complemento a 2**

Complemento a 2

- Se *i* è un numero maggiore di 0, si memorizza la sua rappresentazione in base 2
- Se *i* è un numero minore di 0, allora, anziché memorizzare il numero originale *i*, si memorizza, in base 2, il numero naturale risultante dalla somma algebrica
  $2^N + i$
  dove *N* è il numero di bit su cui si intende memorizzare il numero *i*

Condizioni da rispettare

- Il vincolo da rispettare, affinché si possa correttamente rappresentare un numero *i* negativo, in complemento a 2 su *N* bit, è che il risultato della somma
  $2^N + i$
  - Sia un numero positivo
  - Sia rappresentabile sugli *N* bit di cui si dispone per rappresentare il numero
- Inoltre, per evitare ambiguità nella rappresentazione, i valori che può assumere *i* quando è positivo non devono mai sovrapporsi ai valori possibili che può assumere $2^N + i$ quando *i* è invece negativo

Intervalli di valori 1/3

- Nel complemento a 2 gli intervalli di valori positivi e negativi rappresentabili sono più bilanciati possibile
  - Ossia la lunghezze dei due intervalli sono le più vicine possibili
- In particolare
  - Valori positivi nell'intervallo $[0, 2^{N-1}-1]$
  - Valori negativi nell'intervallo $[-2^{N-1}, -1]$
- Vediamo come si arriva a questa suddivisione

Intervalli di valori 2/3

- Dati i vincoli esposti nella slide 35, e considerando che l'intervallo di numeri naturali che si possono rappresentare su N bit è $[0, 2^N-1]$
- Il modo più bilanciato di suddividere gli intervalli di rappresentabilità su N bit tra numeri positivi e numeri negativi (rappresentati come $2^N + i$) è il seguente
- Utilizzare, per i valori positivi di i, metà dell'intervallo massimo di rappresentabilità dei naturali su N bit
  - Ossia valori positivi nell'intervallo $[0, 2^{N-1}-1]$
- Utilizzare l'altra metà dell'intervallo massimo di rappresentabilità dei naturali su N bit per rappresentare i valori negativi di i, ossia per rappresentare il risultato della somma $2^N + i$

Intervalli di valori 3/3

- Ossia l'intervallo di valori possibili per la somma $2^N + i$ è $[2^{N-1}, 2^N-1]$
- Invertendo la formula $2^N + i$, si ottiene che l'intervallo di valori negativi di i rappresentabili è $[2^{N-1} - 2^N, 2^N-1 - 2^N]$ ossia, eseguendo le sottrazioni, è $[-2^N-1, -1]$
- Mettendo assieme l'intervallo di rappresentabilità dei valori positivi e quello di rappresentabilità dei valori negativi, si ottiene che, mediante rappresentazione in complemento 2, si possono rappresentare, con N bit, tutti i numeri interi nell'intervallo

  $[-2^{N-1}, 2^{N-1}-1]$

Valore del bit più significativo

- Le rappresentazioni in base 2 su *N* bit utilizzate per rappresentare
  1) I numeri positivi in complemento a 2
     - Ossia le rappresentazioni dei numeri naturali nell'intervallo $[0, 2^{N-1}-1]$
     - Hanno tutte il bit *più significativo*, ossia quello in posizione *N-1* (ossia il primo bit da sinistra), uguale a 0
  2) I numeri negativi in complemento a 2
     - Ossia le rappresentazioni dei numeri naturali nell'intervallo successivo, $[2^{N-1}, 2^{N}-1]$
     - hanno tutte il bit più significativo uguale ad 1

Da rappresentazione a valore

- Di conseguenza, se una sequenza di *N* bit
  - è usata per rappresentare un numero intero in complemento a 2
  - rappresenta un valore naturale *n* se interpretata come rappresentazione di un numero naturale
- Allora
  - Se il primo bit è a 0, allora il numero intero *i* rappresentato dalla sequenza è uguale ad *n*
  - Se il primo bit è ad 1, allora il numero intero *i* rappresentato dalla sequenza è uguale a $n-2^N$

Doppia interpretazione

- Quindi una configurazione di *N* bit con il bit più significativo ad 1 rappresenta
  - un valore positivo se interpretata come la rappresentazione di un numero naturale in base 2
  - un valore negativo se interpretata come la rappresentazione di un numero in complemento a 2

Domande

- Quale valore naturale in base 2 è rappresentato dalla configurazione di bit che invece rappresenta il valore -1 in complemento a 2
- E da quella che rappresenta il valore -2^{N-1} in complemento a 2?

Risposte

- $2^{N-1}-1$
- $2^{N}-1$

Vantaggi del complemento a 2

- C'è una sola rappresentazione per lo 0
  - Tutti i bit a 0
- Gli algoritmi di calcolo delle operazioni di somma, sottrazione, moltiplicazione e divisione sono gli stessi dei numeri naturali rappresentati in base 2

Rappresentazione **int**

- Gli oggetti di tipo **int** sono tipicamente rappresentati in complemento a 2

- Adesso dovrebbe esservi più chiaro perché è vero che:

"Ci sono solo 10 tipi di persone al mondo: quelle che conoscono la rappresentazione dei numeri in base 2, e quelle che non la conoscono"

Esercizi

- Completare la quinta esercitazione
- Link alla videoregistrazione:
  https://drive.google.com/file/d/1TKz1dm5x_f5bH9P_h88dl2da1wbt3nfT/view?usp=sharing