Appunti di informatica

Tratti da[1]
Bolchini – Brandolese – Salice – Sciuto, Reti logiche
Dispense di Alessandro Pellegrini
Dispense del corso di Informatica del politecnico di Milano
http://home.dei.polimi.it/amigoni/InformaticaB.html

La codifica dell'informazione è una disciplina molto vasta che vede il suo inizio nella forma rigorosa in cui oggi è conoscuta, agli anni 50, con l'avvento dei primi calcolatori programmabili. È grazie all'opera di Turing, Von Neumann, Shannon[2] che questa disciplina ha mosso i primi passi.

Nell'accezione moderna la codifica dell'informazione è strettamente legata all'elaborazione automatica e molte sono le sfumature che essa assume. A seconda dell'ambito applicativo si parla di codifica numerica, di crittografia di compressione di rilevamento e correzione di errori e molte altre specifiche discipline. *In questi appunti si discutono* i fondamenti della codifica di informazioni di tipo numerico finalizzata alla elaborazione automatica e se ne mostrano i vantaggi, svantaggi e applicazioni nel campo della progettazione di sistemi digitali.

Codici e codifiche

Abbiamo informazioni (numeri, testi, immagini, suoni, etc…) che vogliamo rappresentare (e poter elaborare) in un calcolatore. Per motivi tecnologici, la rappresentazione di queste informazioni deve avvenire utilizzando solo due simboli che corrispondono al valore di grandezze fisiche (tensioni e correnti) con cui vengono azionati i dispositivi elettronici.

L'associazione tra una sequenza di simboli ed un concetto astratto o un oggetto concreto prende il nome di codifica. Il codice è pertanto un insieme di parole in cui ogni parola è costituita da simboli. I simboli utilizzati per costruire le parole vengono detti alfabeto.

1 I testi riportati sono la riproduzione dell'originale. Il testo in corsivo è dell'autore degli appunti.
2 Tra i padri fondatori dell'informatica come disciplina scientifica

Esempio 1

$Alfabeto_1 = \{a, b, c, d, \ldots z\}$

$Codice_1 = dizionario\ delle\ parole\ italiane$

$Codice_2 = dizionario\ delle\ parole\ inglesi$

Esempio 2

$Aalfabeto_1 = \{a, b, c, d, \ldots z, \tilde{n}\}$

$Codice_2 = dizionario\ delle\ parole\ spagnole$

Esempio 3

$Alfabeto_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$Codice_1 = numeri\ naturali\ (\mathbb{N})$

Esempio 4

$Alfabeto_1 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -,\}$

$Codice_1 = numeri\ relativi\ (\mathbb{N})$

È importante notare che mentre un codice può contenere un numero infinito di parole, un alfabeto è sempre costituito da un numero finito di simboli.

La definizione di un codice *può richiedere* anche un insieme di regole che permettono di costruire le parole. Non tutte le sequenze di simboli dell'alfabeto, infatti, appartengono necessariamente al codice. *Nel caso del dizionario delle parole italiane, le parole "aaaaaa" e "bcbvddvdfv" non sono valide.* Ciò nonostante, esistono codici in cui qualsiasi combinazione dei simboli dell'alfabeto è ammissibile, *come nel caso dell'esempio 3*.

Codificare significa stabilire una associazione fra l'insieme delle informazioni e l'insieme dei codici. *Decodificare significa ottenere l'informazione di partenza a partire dal codice che la rappresenta*

Esempio 5

$Informazioni = \{lun, mar, mer, gio, ven, sab, dom\}$

$Alfabeto = \{0, 1\}$

$Codice = \{00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111\}$

$$Codifica_1 = \left\{ \begin{array}{ll} lun & \rightarrow 00 \\ mar & \rightarrow 01 \\ mer & \rightarrow 10 \\ gio & \rightarrow 11 \\ ven & \rightarrow 001 \\ sab & \rightarrow 010 \\ dom & \rightarrow 011 \end{array} \right\} \qquad Codifica_2 = \left\{ \begin{array}{ll} lun & \rightarrow 000 \\ mar & \rightarrow 001 \\ mer & \rightarrow 010 \\ gio & \rightarrow 011 \\ ven & \rightarrow 100 \\ sab & \rightarrow 101 \\ dom & \rightarrow 110 \end{array} \right\}$$

Di particolare interesse sono i codici e le codifiche destinati a rappresentare informazioni di tipo numerico.

Esempi

- Sistemi di numerazione posizionale
- Notazione scientifica
- Frazioni
- Numeri con la virgola

È importantissimo sottolineare che la codifica è una convenzione. Non esiste una codifica universale, esistono diversi soggetti che si mettono d'accordo su come rappresentare le informazioni in modo tale che questa rappresentazione sia comprensibile da tutti i soggetti che la conoscono.

Un [...] criterio per la classificazione dei codici riguarda la lunghezza delle parole: in alcuni casi essa è costante e prefissata *(come nei casi che affronteremo in questi appunti)*, in altri casi parole diverse possono avere lunghezze diverse. *Nel primo caso parliamo di codici a lunghezza fissa, nel secondo caso di codici a lunghezza variabile.* Questi ultimi trovano interessanti applicazioni nel campo della compressione delle informazioni.

Codifica delle informazioni nel sistema binario

Quando l'alfabeto e costituito da soli due simboli, si parla di codifica binaria. L'alfabeto a cui si fa riferimento nell'ambito dell'informatica è costituito dai simboli 0 e 1.

L'unità fondamentale di rappresentazione dell'informazione è il bit. Possiamo pensare ad un bit come ad una qualsiasi entità che può assumere solo due valori. Ad esempio pieno/vuoto, 1/0 ,acceso/spento, verda/giallo, etc.

Esempio

Bianco/Nero

1 bit → 2 informazioni  oppure

2 bit → 4 informazioni 

In generale con L bit rappresento 2^L *combinazioni possibili di questi valori. Rappresentando il valore del bit con i simboli 0 e 1 posso creare codici numerici binari. Le informazioni precedenti diventano 0, 1 – 00, 01, 10, 11. È possibile calcolare il numero minimo di bit da utilizzare per rappresentare N informazioni, considerando che deve essere soddisfatta la relazione*

$$2^L \geq N$$

In termini più semplici, devo prevedere un numero sufficiente di bit che mi permetta di "etichettare" tutte le informazioni che voglio rappresentare e per fare ciò ho bisogno di un numero di combinazioni pari o superiore al numero di informazioni.

Il problema che si vuole affrontare riguarda la creazione di codici rappresentanti informazioni di tipo numerico. In particolare si vuole trovare un modo per rappresentare, mediante stringhe (cioè sequenze) di bit gli insiemi $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ *. Iniziamo da alcune semplici considerazioni:*

1) *gli insiemi numerici definiti in matematica sono infiniti. Significa che per rappresentarli è necessario avere un numero infinito di combinazioni possibili e quindi un numero infinito di bit per i codici*
2) *le informazioni che vogliamo rappresentare devono essere elaborate da dispositivi reali che per loro stessa natura non possono contenere un numero infinito di componenti*

Ne consegue che nella rappresentazione delle informazioni per applicazioni pratiche faremo riferimento ad un numero fisso di bit, che si tradurrà in una limitazione delle informazioni rappresentabili. Ad esempio nel caso si voglia rappresentare con 64 bit l'insieme dei numeri interi, se ne potranno rappresentare solo 2^{64} *di essi. Pur essendo un numero molto grande, non è però un numero*

infinito. Significa cioè che ci sarà sempre almeno un'informazione (in questo caso un numero) che non sarà possibile codificare.

Ci si pone quindi il problema di rappresentare gli insiemi numerici facendo attenzione, per quanto possibile a tre aspetti

1) *trovare una regola generale che permetta, fissato un numero di bit, di generare un codice*
2) *fare in modo che i codici dei numeri rappresentati si comportino come i numeri che rappresentano. Si vorrebbe ciò fare in modo che, ad esempio, la somma dei codici che rappresentano i numeri 2 e 3 sia il codice che rappresenta il numero 5*
3) *trovare una regola che permetta di decodificare una stringa di bit, sapendo che è stata generata mediante un metodo noto*

Sistemi posizionali e numeri naturali (rappresentazione di $\mathbb{N}$)

Possiamo definire informalmente un sistema posizionale come un sistema di codifica che assegna ad ogni simbolo un significato a seconda della posizione che occupa all'interno della parola.

Esempio

$Alfabeto_1 = [0,1,2,3,4,5,6,7,8,9]$

Le cifre 2, 3, 5 utilizzate per comporre il codice 325 nel sistema decimale, viene interpretata come

$$3\times10^2+2\times10^1+5\times10^0$$

Ma le stesse cifre messe in ordine inverso, quindi 523, rappresentano la quantità

$$5\times10^2+2\times10^1+3\times10^0$$

Ogni cifra assume un significato diverso a seconda della posizione che occupa nella sequenza: la cifra rappresenta il coefficiente della potenza del 10 in una determinata posizione. Il numero di simboli utilizzati per creare i codici gioca un ruolo essenziale nella fase di codifica e interpretazione del significato di una data parola. *Esso prende il nome di base del sistema.*

Esempio

Codice	Base	Alfabeto
Binario	2	$\mathcal{A} = \{0,1\}$
Ottale	8	$\mathcal{A} = \{0,1,2,3,4,5,6,7\}$
Decimale	10	$\mathcal{A} = \{0,1,2,3,4,5,6,7,8,9\}$
Esadecimale	16	$\mathcal{A} = \{0,1,2,3,4,5,6,7,8,9,\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D},\mathrm{E},\mathrm{F}\}$

Il significato delle cifre è associato alle potenze del 10 perché nel sistema decimale si usano 10 simboli per scrivere i numeri. I sistemi posizionali hanno però validità generale, pertanto possiamo utilizzare la stessa notazione anche per sistemi che usano solo due simboli. Analogamente a prima, il significato della cifra sarà associato alle potenze del 2 (anziché del 10) perché 2 è il numero di simboli utilizzato!

Esempio

$Alfabeto_1 = \{0, 1\}$

Le cifre 1 e 0 utilizzate per comporre il codice 101 nel sistema binario, viene interpretata come

$$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

ma le stesse cifre messe in ordine diverso, ad esempio 011, rappresentano la quantità

$$0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 3$$

Mediante la notazione posizionale siamo in grado di associare un codice binario ad un numero naturale, semplicemente calcolando il valore numerico associato ad ogni cifra e sommando i valori tra di loro.

Indichiamo con X_c *il codice binario dato.*

Per ottenere il corrispondente numero naturale, è sufficiente eseguire le operazioni viste sopra e ricavare il numero decimale corrispondente

Semplificando, possiamo dire di aver trovato un metodo per decodificare un codice binario e ricavare il numero decimale ad esso associato.

Ci si chiede adesso: dato un numero naturale, qual è il codice che lo rappresenta?

Equivalentemente ci si può chiedere: dato un numero decimale, quali sono i coefficienti delle potenze di 2 che formano il codice corrispondente?

Chiamiamo X *il numero naturale che vogliamo rappresentare.*

La metodologia per il calcolo del codice ad esso associato si basa sull'equivalenza fra il valore del numero decimale e i coefficienti associati alle potenze di 2.

Osservazione

La lunghezza minima del codice deve essere tale da rispettare la relazione $2^L \geq N$

.

Vediamo tramite un esempio, come è possibile calcolare i coefficienti.

X = 13 → numero naturale da codificare

N = 4 → numero di bit usati per la rappresentazione

$X_c = X_1 X_2 X_3 X_4$ → *codice associato a 13, è ciò che vogliamo calcolare*

$X_{1,} X_{2,} X_{3,} X_4$ → *i coefficienti delle potenze di 2 che formano i codici*

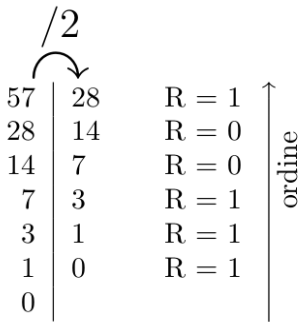
Sappiamo che $X = X_1 \times 2^3 + X_2 \times 2^2 X_3 \times 2^1 + X_4 \times 2^0$ *. Dobbiamo quindi trovare le incognite* $X_{1,} X_{2,} X_3 e\, X_4$ *.*

La regola che permette di ottenere le incognite e quindi di generare il codice, si basa sull'applicazione di alcuni passaggi matematici. Infatti

1) *data l'equazione* $X = X_1 \times 2^3 + X_2 \times 2^2 X_3 \times 2^1 + X_4 \times 2^0$
2) *possiamo dividere dividere entrambi i membri per 2, ottenendo*
    1. $\frac{X}{2} = (X_1 \times 2^2 + X_2 \times 2^1 X_3 \times 2^1 + \frac{X_4}{2})$ → $7.5 = (X_1 \times 2^2 + X_2 \times 2^1 X_3 \times 2^1 + \frac{X_4}{2})$
3) *considero il membro di sinistra composto da parte intera e parte frazionaria:*
    1. *7.5 = 7 + 0.5*
4) *anche il membro di destra può essere raggruppato in una parte intera e una frazionaria*
    1. *Siccome* $X_{1,} X_{2,} X_{3,} X_4$ *possono essere o 0 o 1 possiamo affermare che i primi tre termini compongono la parte intera (in quanto somma di potenze del 2 moltiplicate per 0 o 1), e l'ultimo la parte frazionaria (in quanto* X_4 *potrebbe essere 1 e quindi* $\frac{1}{2} = 0.5$ *)*
    2. $PARTE\ INTERA = X_1 \times 2^2 + X_2 \times 2^1 X_3 \times 2^0$
       $PARTE\ DECIMALE = \frac{X_4}{2}$
5) *uguaglio la parte frazionaria del primo membro con la parte frazionaria del secondo membro*
    1. $0.5 = \frac{X_4}{2}$ → $X_4 = 1$
6) *uguaglio la parte intera del primo membro alla parte intera del secondo membro*
    1. $7 = X_1 \times 2^2 + X_2 \times 2^1 X_3 \times 2^0$

7) *Si è ritornati ad una situazione identica a quella del punto 1). Mi basta quindi applicare lo stesso procedimento, fino a quando non trovo il valore di* X_1

Regola pratica per la conversione della parte intera In generale, per convertire un numero decimale in un'altra base, è sufficiente dividere ripetutamente il numero per la nuova base finché non si ottiene 0 come risultato, e scrivere poi i resti ottenuti, a partire dalla posizione meno significativa. Ad esempio, per convertire il numero 57_{10} in base due è sufficiente dividerlo ripetutamente per due:



Esempio

Utilizzando il sistema posizionale, la somma dei codici corrisonde al codice della somma, infatti

$$\begin{array}{rcr} 110 & + & 6 \\ 10 & = & 2 \\ \hline 1000 & & 8 \end{array}$$

Numeri relativi (rappresentazione di $\mathbb{Q}$)

Rappresentazione in modulo e segno

Una possibile soluzione per la rappresentazione dei numeri relativi consiste nell'uso di una codifica naturale per il valore assoluto e di un simbolo aggiuntivo per la rappresentazione del segno. Convenzionalmente si utilizza il simbolo 0 per indicare valori positivi e 1 per indicare valori negativi. *Vediamo tramite un esempio, come si codificano i numeri negativi. Chiamiamo*

-X = -13 → numero naturale da codificare

X → numero positivo

N = 4 → numero di bit usati per la rappresentazione del numero positivo

$X_c = 1101$ *→ codice di X*

$\bar{X}_c = 11101$ *→ codice di -X*

Il primo bit di $\bar{X}_c$ *corrisponde al segno meno*

Questo metodo prende il nome di rappresentazione in modulo e segno. È un metodo intuitivo e veloce per la rappresentazione dei numeri negativi nel sistema binario. Tuttavia essa presenta dei problemi in quanto:

1) *non essendo puramente posizionale, non sempre si verifica che la somma di due codici corrisponde al codice della somma*
2) *il numero 0 nel sistema decimale è rappresentato due volte*
3) *Non viene sfruttato al massimo il numero di combinazioni possibili per rappresentare i numeri*

Rappresentazione in complemento a 2

La rappresentazione in complemento a 2 ha lo scopo di integrare l'informazione relativa al segno all'interno della parola di codice, contrariamente a quanto avviene nella codifica in modulo e segno in cui, benché adiacenti, segno e valore numerico sono *concettualmente separati*.

Per poter fare in modo che i codici dei numeri negativi mantengano le proprietà matematiche della somma, è necessario generare tali codici imponendo il rispetto di una regola matematica. Consideriamo

$-X \rightarrow$ *numero da convertire*
$N \rightarrow$ *numero di bit da usare per la rappresentazione*
$X_c \rightarrow$ *codice binario naturale del numero senza segno*
$\bar{X}_c \rightarrow$ *codice in complemento a due del numero negativo*

La regola che permette di generare il codice del numero negativo è la seguente:

$X_c + \bar{X}_c = (2^N)_2 \rightarrow$ *regola generatrice del codice*

Esempio

-10	$\rightarrow$ *numero da convertire*
$N = 8$	$\rightarrow$ *numero di bit da usare per la rappresentazione*
$X_c = 00001010$	$\rightarrow$ *codice binario naturale del numero senza segno*
$\bar{X}_c$	$\rightarrow$ *codice in complemento a due del numero negativo . è ciò che vogliamo calcolare*
$2^N = 128$	
$(2^N)_2 = 100000000$	*sarà lungo* $N+1$ *bit*
$X_c + \bar{X}_c = (2^N)_2$	$\rightarrow 00001010 + \bar{X}_c = 100000000$

Per trovare la codifica in complemento a due, risolvo l'equazione

$$\begin{array}{rcl} 00001010 & + & \\ \bar{X}_c & = & \Rightarrow \quad \bar{X}_c = 11110110 \\ 100000000 & & \end{array}$$

Regola pratica per la codifica

Converti il numero senza segno in binario naturale su N bit . A partire dal bit meno significativo , lascia invariati tutti i bit fino al primo 1 incluso e inverti tutti i bit successivi

Regola pratica per la decodifica

Se il codice inizia per 0 converti il codice come se fosse binario naturale .
Se il codice inizia per 0 converti il codice come se fosse binario naturale e poi sottrai 2^N

Numeri razionali (rappresentazione di $\mathbb{Z}$)

A differenza di quanto accade per i numeri interi (naturali e relativi), in un certo un intervallo [a,b] la quantità di numeri razionali è infinita. Per questa ragione, servirebbero infiniti bit anche per rappresentare i numeri razionali compresi fra un minimo e un massimo.

Esempio

- *Considerando i numeri naturali, fra 0 e 5 esistono 4 numeri e l'insieme è formato da 6 numeri (0,1,2,3,4,5)*
- *Considerando i numeri relativi, fra -4 e 3 esistono 6 numeri e l'insieme è formato da 8 numeri (-4, -3,-2,-1,0,1,2,3)*
- *Considerando i numeri razionali, fra 0 e 1 esistono infiniti numeri (0.1, 0.001, 0.0001, 0.2, 0.002, 0.002, etc…)*

Questo significa che, fissato un numero di bit, è possibile rappresentare solo alcuni dei numeri razionali di un dato intervallo.

Esempio

N = 5 bit → 32 possibili combinazioni → posso rappresentare solo 32 numeri razionali in un intervallo. Considerando i numeri razionali compresi tra 0 e 1, solo 32 di essi saranno rappresentabili!

Osservazione

Nel sistema decimale i numeri razionali possono essere rappresentati come rapporto (frazioni) o come numeri separati da una virgola che permette di capire qual è la parte intera e qual è la parte frazionaria. Nel caso di numeri periodici, l'utilizzo della notazione con la virgola non permette di scrivere tutte le cifre decimali (che sono comunque infinite) e per questo motivo si usa una notazione diversa ($1{,}\overline{3}$ significa un 1 seguito da infiniti 3). In questo caso, se usiamo un numero finito di cifre per la parte frazionaria stiamo facendo un'approssimazione del numero periodico.

Come già fatto per i numeri interi, possiamo sfruttare la notazione posizionale per interpretare un numero binario "con la virgola"

Esempio

Sapendo che il primo bit a sinistra del codice si riferisce a 2^0 *0101 nel sistema binario, la sequenza viene interpretata come*

$$0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}=0.75$$

Sapendo che il primo bit a sinistra del codice 1011, si riferisce a 2^1, la sequenza viene interpretata come

$$1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2}=2.75$$

Ci si chiede adesso: dato un numero frazionario, qual è il codice che lo rappresenta?

Equivalentemente ci si può chiedere: dato un numero frazionario, quali sono i coefficienti delle potenze di 2 che formano il codice corrispondente?

Chiamiamo

- X *= 13.71 → numero frazionario da codificare*
- X_I *= 13 → parte intera*
- X_F *= 0.71 → parte frazionaria*
- N_I *= 4 → numero di bit usati per la rappresentazione della parte intera*
- N_F *= 5 → numero di bit usati per la rappresentazione della parte decimale*
- $N=N_I+N_F$ *→ numero di bit totali usati per la rappresentazione*

Possiamo scrivere che $X=X_I+X_F$ *, infatti 13.71 = 13 + 0.71*

Un numero con la virgola può essere quindi visto come la somma di un numero intero ed un numero frazionario compreso tra 0 e 1.

Abbiamo visto come codificare i numeri interi, quindi la prima parte del codice si potrà ottenere con i metodi già studiati.

Il codice di 13, rappresentato su 4 bit è X_{IC} *= 1101.*

Rimane da capire come codificare la parte frazionaria.

Chiamiamo X_{DC} *→ il codice della parte frazionaria, ciò che volgiamo trovare.*

- $X_{DC}=X_{1,}X_{2,}X_{3,}X_4X_5$ *→ i coefficienti delle potenze di 2 che formano i codici*

Sappiamo che $X_D = X_1 \times 2^{-1} + X_2 \times 2^{-2} X_3 \times 2^{-3} + X_4 \times 2^{-4} + X_5 \times 2^{-5}$ *. Dobbiamo quindi trovare le incognite* $X_1, X_2, X_3, X_4 e X_5$ *.*

La regola che permette di ottenere le incognite e quindi di generare il codice, si basa sull'applicazione di alcuni passaggi matematici. Infatti

1. *data l'equazione* $X_D = X_1 \times 2^{-1} + X_2 \times 2^{-2} X_3 \times 2^{-3} + X_4 \times 2^{-4} + X_5 \times 2^{-5}$
2. *moltiplicare entrambi i membri per 2, ottenendo*
    1. $2 \times X_D = 2 \times \left(X_1 \times 2^{-1} + X_2 \times 2^{-2} X_3 \times 2^{-3} + X_4 \times 2^{-4} + X_5 \times 2^{-5} \right) \rightarrow$
    $1.42 = X_1 + X_2 \times 2^{-1} X_3 \times 2^{-2} + X_4 \times 2^{-3} + X_5 \times 2^{-4}$
3. *considero il membro di sinistra composta da parte intera e parte decimale:*
    1. *1.42 = 1 + 0.42*
4. *il membro di destra può essere raggruppato in una parte intera e una frazionaria*
    1. *Siccome* X_1, X_2, X_3, X_4, X_5 *possono essere o 0 o 1 possiamo affermare che* X_1 *è la parte intera (in quanto può valere o 0 o 1), e la restante somma è la parte frazionaria (in quanto* X_2, X_3, X_4, X_5 *vengono divisi per le potenze di 2)*
    2. $PARTE\ INTERA = X_1$
    $PARTE\ DECIMALE = X_2 \times 2^{-1} X_3 \times 2^{-2} + X_4 \times 2^{-3} + X_5 \times 2^{-4}$
5. *uguaglio la parte decimale del primo membro con la parte decimale del secondo membro*
    1. $1 = X_1 \rightarrow X_1 = 1$
6. *uguaglio la parte intera del primo membro alla parte intera del secondo membro*
    1. $0.42 = X_2 \times 2^{-1} X_3 \times 2^{-2} + X_4 \times 2^{-3} + X_5 \times 2^{-4}$
7. Sono ritornato ad una situazione identica a quella del punto 1). Mi basta quindi applicare gli stessi passaggi, fino a quando non trovo il valore di X_5

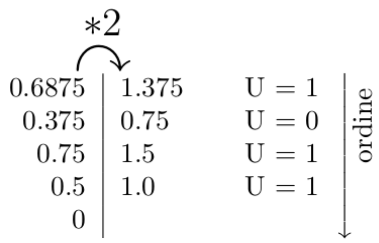
Questo tipo di codifica prende il nome di codifica a virgola fissa perché si stabilisce a priori quanti bit vengono usati per la parte intera e quanti per la parte frazionaria

Regola pratica per la conversione della parte frazionaria Per convertire la parte frazionaria di un numero decimale in binario, è sufficiente:

- moltiplicare la parte frazionaria per 2
- scrivere il valore della parte intera ottenuta
- ripetere il procedimento sulla nuova parte frazionaria ottenuta, finché non si ottiene una parte frazionaria nulla
- Prendere le parti intere dalla più significativa alla meno significativa

Nota: nel caso dei numeri periodici, si potrebbe non ottenere mai una parte frazionaria nulla.

Ad esempio, per convertire 0.6875_{10} in binario:



Esercizi di codifica

Rappresentazione di numeri interi,

Generare il codice binario dei seguenti numeri interi decimali, calcolando prima il numero di bit necessari: 2, 4, 8, 16, 30, 65, 130

Rappresentazione di numeri relativi

Generare il codice binario dei seguenti numeri interi decimali negativi, utilizzando per ognuno il numero di bit indicato tra parentesi: -2 (4), -4 (8), -4 (4), -8 (5)

Rappresentazione di numeri frazionari

Generare il codice binario dei seguenti numeri frazionari, utilizzando per la parte intera e per la parte frazionaria di ognuno il numero di bit indicato tra parentesi

12.5 [N_I = 4, N_D = 6]

14.75 [N_I = 5, N_D = 5]

31.015625 [N_I = 6, N_D = 7]

52.44 [N_I = 7, N_D = 5]

Esercizi di decodifica

Decodifica di numeri interi,

Decodificare i seguenti codici binari, calcolando il numero intero che rappresentano: 10001, 1100111, 1110, 000111, 11101

Decodifica di numeri relativi

Decodificare i seguenti codici binari ottenuti mediante complemento a 2, calcolando il numero relativo che rappresentano: 10001, 1100111, 0110, 00111, 11101

Rappresentazione di numeri frazionari

Decodificare i seguenti codici binari che rappresentano numeri frazionari. Il numero di bit utilizzati per la parte intera e per la parte frazionaria sono indicati tra parentesi

1000111011 [N_I = 4, N_D = 6]

1000111011 [N_I = 5, N_D = 5]

1011110111011 [N_I = 6, N_D = 7]

100010001001 [N_I = 7, N_D = 5]

Esercizi sulla somma

Sapendo che i seguenti codici binari sono stati ottenuti mediante la tecnica del complemento a 2, eseguire in binario le seguenti somme e verificare il risultato decodificando i codici.

00001111 + 10001100

10000110 + 10011100

00001111 + 10011100

Spunti di riflessione

1) Dati N bit, il numero massimo di combinazioni possibili è 2^N . esistono tuttavia metodi di codifica che riescono a rappresentare più di 2^N valori (come ad esempio la codifica a virgola mobile, che non è stata affrontata).
2) La codifica modulo e segno permette di eseguire somme e sottrazioni ma bisogna gestire la presenza del segno come informazione codificata a parte
3) I calcoli effettuati da un computer possono essere soggetti ad errori se non si tiene conto che i numeri in esso rappresentati possono essere approssimati. Aumentare il numero di bit per la rappresentazione delle informazioni numeriche, diminuisce l'errore di calcolo ma non lo elimina completamente
4) All'interno di un calcolatore esistono circuiti che effettuano le operazioni matematiche fra numeri. La scelta della rappresentazione influisce quindi sul tipo di circuito da utilizzare.
5) Per ogni tipo di codifica esistono dei limiti di rappresentazione, ossia un minimo ed un massimo rappresentabile in base al numero di bit utilizzati (N). Nel caso della rappresentazione a virgola fissa va poi tenuto in considerazione che non tutti i numeri frazionari possono essere rappresentati in maniera esatta. I limiti di rappresentazione per i numeri interi e relativi, in base alla rappresentazione sono
    1. da 0 a 2^N-1 per i numeri naturali
    2. da $-2^{N-1}+1$ a $2^{N-1}-1$ per i numeri relativi (modulo e segno)
    3. da -2^{N-1} a $2^{N-1}-1$ per i numeri relativi (complemento a due)
6) I numeri frazionari negativi si rappresentano utilizzando il complemento a 2 per la parte intera. Ad esempio, il numero -5.25 si tratta come se fosse -6 + 0.75. Si codifica il -6 in complemento a 2 e lo 0.75 con la tecnica studiata precedentemente