La Rappresentazione dell'Informazione

Prof.Ing.S.Cavalieri

Codifica dell'Informazione

- Un sistema numerico è determinato da:
  - Un insieme finito di cifre (simboli)
  - Un insieme finito di regole:
    - ✓Non posizionali: il valore di ciascuna cifra nella rappresentazione è indipendente dalla sua posizione (es. M, C nei numeri romani)
      - CX = 110
      - MMC=2100
    - ✓Posizionali: ad ogni posizione della cifra all'interno della rappresentazione è associato un peso

- Verranno considerati solo i sistemi di codifica posizionali

Codifica di tipo Posizionale

- Il numero è rappresentato da una sequenza di **N** cifre:

$$d_{N-1}\, d_{N-2}\, d_{N-3}\, d_1\, d_0$$

- Ogni cifra ha un peso diverso a seconda della posizione (i) che occupa
  - detta b la base della rappresentazione, il peso è dato da b^i
- Il valore numerico si ottiene sommando i prodotti delle N cifre per il rispettivo peso.

$$d_{N-1} \bullet b^{N-1} + d_{N-2} \bullet b^{N-2} + \ldots\ldots\ldots + d_2 \bullet b^2 + d_1 \bullet b^1 + d_0 \bullet b^0$$

  - ad esempio nel sistema decimale (base b=10)

$$10.061.974 = 1x10^7 + 0x10^6 + 0x10^5 + 6x10^4 + 1x10^3 + 9x10^2 + 7x10^1 + 4x10^0$$

Codifica nei Calcolatori

- Qualunque rappresentazione utilizzata in un calcolatore è basata sull'utilizzo di cifre binarie (bit=0,1) e sequenze di **N bits**
  - N multiplo di 8, tipicamente N=8 bit, 16 bit, 32 bit, 64 bit, 80 bit
  - 8 bits = byte

- Le principali modalità di rappresentazione utilizzate nei calcolatori sono:
  - Numeri interi senza segno (Naturali) e Relativi
  - Numeri razionali
  - Caratteri

Codifica Numeri Naturali

- I numeri interi senza segno, ossia i numeri ***Naturali incluso lo 0***, vengono rappresentati su N bits attraverso un algoritmo di conversione

- Ovviamente esiste un limite sul numero Naturale massimo rappresentabile, in dipendenza del numero N di bits

- Chi sceglie N ? In programmazione sarete voi a farlo !

Codifica Numeri Naturali

Primo Passo

- L'algoritmo che converte un numero Naturale in binario è:
  - Si divide il numero da convertire per 2
  - Si riporta il risultato della divisione ed il resto della divisione
  - Si ripete la divisione sul risultato fino a quando il risultato diviene 0
  - Il numero binario è costituito da tutti i resti presi da quello ottenuto per ultimo a quello ottenuto per primo

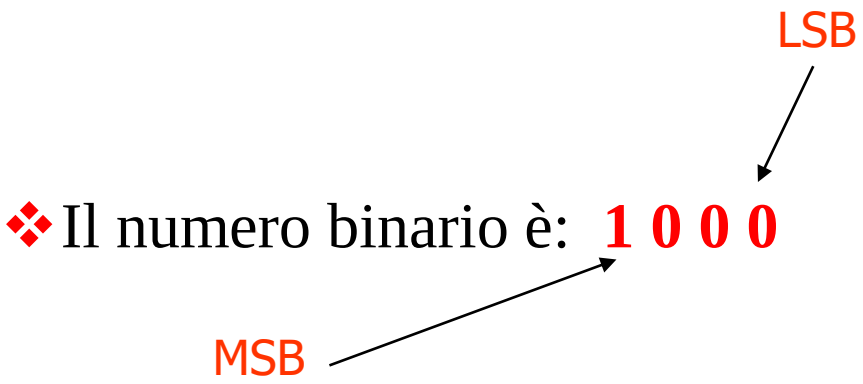
- Esempio: si converta in binario il numero 8

8:2= 4 resto 0 Bit meno significativo (LSB)

4:2= 2 resto 0

2:2= 1 resto 0

1:2= 0 resto 1 Bit più significativo (MSB)



Codifica Numeri Naturali

- Esempio: si converta in binario il numero 71

71:2= 35	resto 1	Bit meno significativo (LSB)
35:2= 17	resto 1	
17:2= 8	resto 1	
8:2= 4	resto 0	
4:2= 2	resto 0	
2:2 = 1	resto 0	
1:2 = 0	resto 1	Bit più significativo (MSB)

- Il numero binario è: **1 0 0 0 1 1 1**

Codifica Numeri Naturali

Secondo Passo:

- Dato un numero intero e ottenuta la sua codifica in binario, il numero di bit ottenuti deve essere riportato al numero **N di bits** che si è scelto per la rappresentazione
- Si inseriscono zeri nelle posizioni più significative (a sinistra)
  - Esempio N=8 bits e numero da convertire 8
  - Sequenza binaria: 1000
  - Si aggiungono 4 bits 0 a sinistra: **0000**1000
- Perché a sinistra e non a destra ?

Codifica Numeri Naturali

- Ad esempio, si supponga di voler rappresentare il numero 8 usando: 1, 2 e 4 bytes
  - La codifica binaria di 8 è 1000
  - La rappresentazione su un byte è 00001000
  - La rappresentazione su 2 bytes è 00000000 00001000
  - La rappresentazione su 4 bytes è 00000000 00000000 00000000 00001000
- Ad esempio, si supponga di voler rappresentare il numero 71 usando: 1, 2 e 4 bytes
  - La codifica binaria di 71 è 1000111
  - La rappresentazione su un byte è 01000111
  - La rappresentazione su 2 bytes è 00000000 01000111
  - La rappresentazione su 4 bytes è 00000000 00000000 00000000 01000111

Codifica Numeri Naturali

- Ad esempio, si supponga di voler rappresentare il numero 350 usando: 1, 2 e 4 bytes
  - La codifica binaria di 350 è 101011110
  - La rappresentazione su un byte NON ESISTE
  - La rappresentazione su 2 bytes è 00000001 01011110
  - La rappresentazione su 4 bytes è
    00000000 00000000 00000001 01011110

Codifica Numeri Naturali

- **La conversione da binario ad intero**, si effettua partendo dal bit meno significativo a quello più significativo, moltiplicando ogni bit per 2 elevato alla posizione del bit:

$$d_{N-1} \bullet 2^{N-1} + d_{N-2} \bullet 2^{N-2} + d_{N-3} \bullet 2^{N-3} + ... + d_1 \bullet 2^1 + d_0 \bullet 2^0$$

  - Esempio (N=16):

$$1000101011010101$$

  - Il numero intero è:

$$1x2^{15} + 1x2^{11} + 1x2^9 + 1x2^7 + 1x2^6 + 1x2^4 + 1x2^2 + 1x2^0 = 35.541$$

Codifica Numeri Naturali

- Il più piccolo numero codificabile è:

$$0 \bullet 2^{N-1} + 0 \bullet 2^{N-2} + 0 \bullet 2^{N-3} + ... + 0 \bullet 2^{1} + 0 \bullet 2^{0} = 0$$

- Il più grande numero codificabile è:

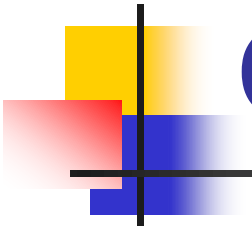
$$1 \bullet 2^{N-1} + 1 \bullet 2^{N-2} + 1 \bullet 2^{N-3} + ... + 1 \bullet 2^{1} + 1 \bullet 2^{0} = 2^{N} - 1$$

- Se N=8, 0-255
- Se N=16, 0-65.535
- Se N=32, 0-4.294.967.295
- Se N=64, 0-18.446.744.073.709.551.615

Codifica in base 16

- La codifica esadecimale viene a volte usata al posto della binaria per ridurre spazio
- Il numero binario viene suddiviso in blocchi di 4 bits a partire dal meno significativo
- Ad ogni gruppo viene sostituito il simbolo esadecimale corrispondete

Decimale	Binario	Esadecimale
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F



Codifica in base 16

- Ad esempio: 1000 1010 1101 0101
  - 0101->5
  - 1101->D
  - 1010->A
  - 1000->8
- Diventa: 8AD5

Codifica Numeri Interi Relativi

- Per i numeri interi relativi si utilizza in genere la:
  - Rappresentazione in Complemento a 2
  - Tale rappresentazione utilizza una particolare operazione binaria: Complemento a 2

Codifica Numeri Interi Relativi

- **Cosa è il complemento a 2 di un numero binario ?**
- Dato un numero binario di N bit, il complemento a 2 di tale numero si ottiene tramite il seguente algoritmo:
  - si procede dal bit meno significativo verso quello più significativo **(da destra a sinistra)**
  - se si incontrano bit 0, essi vengono lasciati inalterati
  - se si incontra il primo bit 1 anche esso viene lasciato inalterato
  - tutti i bit successivi al primo bit 1, vengono invertiti (0 diviene 1, e viceversa)
  - fino a quando si arriva a fine sequenza (a sinistra)

Esempio di Complemento a 2

Esempio: si determini il complemento a 2 del numero 00010100.

- Tutti i bit 0 a partire dal bit meno significativo sono lasciati inalterati e così anche il primo bit 1.
- Tutti gli altri bit vengono invertiti, ottenendo: 11101100.

Esempio: si determini il complemento a 2 del numero 01101001.

- In questo caso non esistono bit 0 a partire dal bit meno significativo.
- Sono il primo bit 1 viene lasciato inalterato.
- Gli altri vengono invertiti, ottenendo: 10010111.

Codifica Numeri Interi Relativi

La rappresentazione in Complemento a 2 di un numero intero relativo su N bit, si effettua nella seguente maniera:

- Per i numeri interi positivi (incluso lo zero): Si applica la regola di conversione in binario ad una sequenza di N-1 bits ed imponendo il bit MSB (quello a destra) pari a 0.

- Per i numeri interi negativi: Si applica una regola basata sull'operazione di complemento a 2 ad una sequenza di N bits

Codifica Numeri Interi Relativi

- I numeri interi positivi (incluso lo zero) sono rappresentati codificando il numero in N-1 bit e imponendo il MSB = 0
  - **MSB, pari a 0** e N-1 bit per la codifica

Esempio: si voglia convertire il numero +5 con 8 bit

- Codifica binaria di 5 = 101
- La sequenza di N-1 (7) bit si ottiene, aggiungendo zeri a sinistra; si ottiene 0000**101**.
- Il primo bit (quello più significativo) viene posto a **0** perchè il numero è positivo
- Dunque il numero binario che rappresenta +5 è: **0**0000**101**

Codifica Numeri Interi Relativi

- I numeri interi negativi sono rappresentati:
  - Si considera la codifica su N bit del valore assoluto
  - Si realizza il complemento a 2 della codifica binaria ottenuta

Esempio: si voglia convertire il numero -5 con 8 bit

- Codifica del numero 5 (valore assoluto) su 8 bit: 00000**101**
- Complemento a 2 del numero ottenuto: 11111011

Codifica Numeri Interi Relativi

Esempio: si voglia convertire il numero 1 con 8 bit
Essendo il numero positivo:
•Codifica binaria di 1: **1**
•Codifica binaria su 7 bit di 1: **0000001**
•Segno **0**
•Codifica: **0**0000001

Esempio: si voglia convertire il numero -1 con 8 bit
Essendo il numero negativo:
•Codifica binaria del valore assoluto (1) su 8 bits 00000001
•Complemento a 2 è 11111111.

Codifica Numeri Interi Relativi

- Regola Generale:
  - I numeri binari che hanno il bit d_{N-1} =0 (MSB) sono positivi
  - I numeri binari che hanno il bit d_{N-1} =1 (MSB) sono negativi
- Dato un numero in complemento a 2, la sua conversione in decimale deve avvenire tramite la formula:

$$-d_{N-1} \bullet 2^{N-1} + d_{N-2} \bullet 2^{N-2} + d_{N-3} \bullet 2^{N-3} + ... + d_1 \bullet 2^1 + d_0 \bullet 2^0$$

Codifica Numeri Interi Relativi

$$-d_{N-1} \bullet 2^{N-1} + d_{N-2} \bullet 2^{N-2} + d_{N-3} \bullet 2^{N-3} + ... + d_1 \bullet 2^1 + d_0 \bullet 2^0$$

- Da questa formula si vede che il numero più piccolo che può essere rappresentato con N bit è:

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

- mentre il numero più grande è:

$$2^{N-2} + 2^{N-3} + ... + 2^1 + 2^0 = 2^{N-1} - 1$$

Codifica Numeri Interi Relativi

Dunque se dispongo di N bit, è possibile rappresentare numeri interi relativi il cui intervallo sarà:

$$-(2^{N-1}).....+(2^{N-1}-1)$$

- Se N=8 ---> -128,….,0,….,+127
- Se N=16 ---> -32.768,….,0,….,+32.767
- Se N=32 ---> -2.147.483.648,….,0,….,+2.147.483.647
- Se N=64 --->-9.223.372.036.854.775.808,…,0,…,+ 9.223.372.036.854.775.807

Codifica Numeri Interi Relativi

Esempio: si voglia convertire il numero binario in complemento a 2 00000001

$$-d_7 \cdot 2^7 + d_6 \cdot 2^6 + d_5 \cdot 2^5 + ... + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

Applicando la formula precedente , si ottiene che il numero decimale è 1.

Esempio: si voglia convertire il numero binario in complemento a 2 11111111

Applicando la formula

$$-d_7 \cdot 2^7 + d_6 \cdot 2^6 + d_5 \cdot 2^5 + ... + d_1 \cdot 2^1 + d_0 \cdot 2^0$$

si ottiene che il numero decimale è dato dalla somma:

$-1*2^7+1*2^6+1*2^5+1*2^4+1*2^3+1*2^2+1*2^1+1*2^0=$

-128+64+32+16+8+4+2+1 =

-128+127=-1

Codifica Numeri Interi Relativi

Perchè si usa la Rappresentazione in Complemento a 2 ?

- Il motivo più rilevante è relativo ai vantaggi ottenibili nell'esecuzione di operazioni elementari come la somma e la sottrazione.
- Queste due operazioni sono quelle che vengono più frequentemente realizzate in un computer, e, dunque, un risparmio nel tempo necessario alla loro esecuzione comporta un indiscusso aumento delle prestazioni di un computer.
- La codifica in complemento a 2 permette un notevole risparmio di tempo nell'esecuzione di somme e sottrazioni

Addizione

Le regole per realizzare l'addizione tra numeri binari:

- 0+0=0 con riporto di 0
- 0+1=1 con riporto di 0
- 1+0=1 con riporto di 0
- 1+1=0 con riporto di 1

Esempio: si vogliano sommare i numeri 00000001 (1) e 00001010(10).

		0	0	0	0	0	0	0	1	+	1
		0	0	0	0	1	0	1	0	=	10
RISULTATO		0	0	0	0	1	0	1	1		11
RIPORTO	0	0	0	0	0	0	0	0			

Addizione

Le regole per realizzare l'addizione tra numeri binari:

- 0+0=0 con riporto di 0
- 0+1=1 con riporto di 0
- 1+0=1 con riporto di 0
- 1+1=0 con riporto di 1

Esempio: si vogliano sommare i numeri 00000011 (3) e 00001010(10).

		0	0	0	0	0	0	1	1	+	3
		0	0	0	0	1	0	1	0	=	10
RISULTATO		0	0	0	0	1	1	0	1		13
RIPORTO	0	0	0	0	0	0	1	0			

Addizione

Le regole per realizzare l'addizione tra numeri binari:

- 0+0=0 con riporto di 0
- 0+1=1 con riporto di 0
- 1+0=1 con riporto di 0
- 1+1=0 con riporto di 1

Esempio: si vogliano sommare i numeri 00000111 (7) e 00001010(10).

		0	0	0	0	0	1	1	1	+	7
		0	0	0	0	1	0	1	0	=	10
RISULTATO		0	0	0	1	0	0	0	1		17
RIPORTO	0	0	0	0	1	1	1	0			

Somme e Sottrazioni

- Dati due numeri, le operazioni di somma o di sottrazione dipendono dai segni
- Se i segni sono gli stessi:
  - Si considerano i numeri in valore assoluto
  - Si sommano tali numeri
  - Il numero risultante sarà ottenuto aggiungendo il segno al numero ottenuto dalla somma.
- Se i segni dei due numeri sono diversi:
  - Si considerano i numeri in valore assoluto
  - Si sottrae il numero più piccolo in valore assoluto dal numero più grande
  - Il numero risultante sarà ottenuto aggiungendo al numero ottenuto il bit di segno del numero in valore assoluto più grande.

Somme e Sottrazioni in Complemento a 2

- **Dati due numeri binari in complemento a due, si applicano le regole dell'addizione a tutti i bit compreso il bit di segno.**

Esempio: Si sommino i numeri a 8 bit 00000010 (+2) e 11111010 (-6).

		0	0	0	0	0	0	1	0	+	+2
		1	1	1	1	1	0	1	0	=	-6
RISULTATO		1	1	1	1	1	1	0	0		-4
RIPORTO	0	0	0	0	0	0	1	0			

Il numero binario risultante è già il risultato con il segno giusto.

Somme e Sottrazioni in Complemento a 2

Esempio: Si sommino i numeri in complemento a 2:
00001100 (+12) e 11100000 (-32).

		0	0	0	0	1	1	0	0	+	+12
		1	1	1	0	0	0	0	0	=	-32
RISULTATO		1	1	1	0	1	1	0	0		-20
RIPORTO	0	0	0	0	0	0	0	0			

Il numero binario risultante è già il risultato con il segno giusto.

Codifica Numeri Interi Relativi

Perchè la Rappresentazione in Complemento a 2 è assai conveniente ?

Da quanto detto è evidente la semplificazione nel calcolo della somma e della sottrazione dei numeri rappresentati in complemento a due.

Riconoscimento Automatico di un Risultato Corretto

- Durante le operazioni di somma in complemento a 2, può succedere:
  - Riporto a monte del bit più significativo. A volte il riporto è pari a 1.
  - Risultato non valido. Generalmente ciò accade perché è possibile che il numero che si ottiene è incompatibile con il numero di bit usati (troppo grande o troppo piccolo): **OVERFLOW**
- Il computer deve riconoscere in modo automatico se il risultato è valido o meno, qualunque cosa accada !

Riconoscimento Automatico di un Risultato Corretto

- Si supponga di lavorare con codifica complemento a 2 su 8 bit (-128,…,+127)
- si consideri la seguente operazione di somma in complemento a due:

  10010000 (-112) e 11110000 (-16).
- Eseguendo la somma, il calcolatore ottiene:

		1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**+**	**-112**
		1	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**=**	**-16**
RISULTATO		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**		**-128**
RIPORTO	1	1	1	1	1	0	0	0	0		

- Si ha un riporto a monte, pari a 1
- Il risultato è 10000000 (–128), che è corretto
- **Conclusione: RISULTATO CORRETTO e si ha Riporto 1**

Riconoscimento Automatico di un Risultato Corretto

- Si supponga di lavorare sempre con codifica complemento a 2 su 8 bit (-128,...,+127)
- Si consideri la seguente operazione di somma in complemento a due:

		1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**+**	**-112**
		1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**=**	**-32**
RISULTATO		**0**	**1**	**1**	**1**	**0**	**0**	**0**	**0**		**+112**
RIPORTO	1	0	0	0	0	0	0	0			

- **Si ha un riporto di 1 a monte del bit più significativo**
- Il risultato è 01110000 (+112), che è errato.
- Che è successo ? Overflow, perché il vero risultato (-144) non si può codificare su 8 bit !
- **Conclusione: RISULTATO ERRATO e si ha Riporto 1**

Riconoscimento Automatico di un Risultato Corretto

- Si supponga sempre di lavorare ad 8 bit (-128,….,+127)
- Si consideri la seguente operazione di somma in complemento a due:
  01111110 (+126) e 00000011 (+3).
- Eseguendo la somma, il calcolatore ottiene:

		0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**+**	**+126**
		0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**=**	**+3**
RISULTATO		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**		**-127**
RIPORTO	0	1	1	1	1	1	1	0			

- **NON si ha un riporto di 1 a monte del bit più significativo**
- Il risultato è -127, che rappresenta palesemente un risultato errato.
- Il vero risultato (+129) non si può codificare su 8 bit ! Overflow !
- **Conclusione: RISULTATO ERRATO e NON si ha Riporto di 1**

Riconoscimento Automatico di un Risultato Corretto

- Il calcolatore deve capire se il risultato che è stato ottenuto sia valido o meno, appena ha calcolato lo stesso risultato.

- Senza ragionamenti: decisione immediata

Riconoscimento Automatico di un Risultato Corretto

- Siano X e Y i due numeri in complemento a due da sommare
- Sia S il risultato ottenuto
- Alla fine dell'operazione di somma, il risultato S viene considerato non valido, se i bit più significativi di X e Y sono uguali e il bit più significativo di S è diverso da essi.

Riconoscimento Automatico di un Risultato Corretto

Esempio: Siano dati i numeri a 8 bit 00100000 (+32) e 10100000 (-96).

X		**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**+**	**+32**
Y		**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**=**	**-96**
S		**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**		**-64**
RIPORTO	0	0	1	0	0	0	0	0			

- il risultato S **è valido**, perché i bit più significativi di X e Y sono diversi.

Riconoscimento Automatico di un Risultato Corretto

Esempio: Siano dati i numeri a 8 bit 01111110 (+126) e 00000011 (+3). Somma impossibile su 8 bit perché il massimo intero positivo rappresentabile è +127

X		**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**+**	**+126**
Y		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**=**	**+3**
S		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**		**-127**
RIPORTO	0	1	1	1	1	1	1	0			

- il risultato S **NON è valido**, perché i bit più significativi di X e Y sono uguali e il bit più significativo di S NON è uguale a loro.

Riconoscimento Automatico di un Risultato Corretto

- Esempio: si consideri la seguente operazione di somma in complemento a due:

  10010000 (-112) e 11110000 (-16).

- Eseguendo la somma, il calcolatore ottiene:

X		**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**+**	**-112**
Y		**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**=**	**-16**
S		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**		**-128**
RIPORTO	1	1	1	1	1	0	0	0	0		

- il risultato S **è valido**, perché i bit più significativi di X e Y sono uguali e il risultato ha il bit più significativo uguale ad essi.

Riconoscimento Automatico di un Risultato Corretto

- Esiste un altro modo alternativo per capire se il risultato della somma in complemento a 2 è valido
- Si considerino gli ultimi due riporti ottenuti, quelli relativi alla somma dei due bit più a sinistra
- Se gli ultimi due riporti sono uguali allora il risultato è valido, altrimenti non lo è

Riconoscimento Automatico di un Risultato Corretto

OK

		1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**+**	**-112**
		1	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**=**	**-16**
		1	**0**	**0**	**0**	**0**	**0**	**0**	**0**		**-128**
RIPORTO	**1**	**1**	1	1	0	0	0	0			

NO

		1	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**+**	**-112**
		1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**=**	**-32**
		0	**1**	**1**	**1**	**0**	**0**	**0**	**0**		**+112**
RIPORTO	**1**	**0**	0	0	0	0	0	0			

NO

		0	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**+**	**+126**
		0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**=**	**+3**
		1	**0**	**0**	**0**	**0**	**0**	**0**	**1**		**-127**
RIPORTO	**0**	**1**	1	1	1	1	1	0			

Riconoscimento Automatico di un Risultato Corretto

OK

X		**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**+**	**+32**
Y		**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**=**	**-96**
S		**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**		**-64**
RIPORTO	**0**	**0**	1	0	0	0	0	0			

NO

X		**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**+**	**+126**
Y		**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**=**	**+3**
S		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**		**-127**
RIPORTO	**0**	**1**	1	1	1	1	1	0			

OK

X		**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**+**	**-112**
Y		**1**	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**=**	**-16**
S		**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**		**-128**
RIPORTO	**1**	**1**	1	1	0	0	0	0			

Codifica dei Numeri Reali

I numeri reali possono essere rappresentati in due modalità:

- Virgola fissa
- Virgola mobile, più diffusa (utilizza la codifica in virgola fissa)
- In ogni caso non è possibile rappresentate tutti i numeri reali nell'intervallo [min, max] a differenza dei numeri interi

Codifica dei Numeri Reali Virgola Fissa

- La rappresentazione in virgola fissa consiste nel rappresentare un numero reale con segno tramite N bit, supponendo fissa la posizione della virgola. Esempio: **1010**.**101**
- In un numero rappresentato in virgola fissa a N bits, viene utilizzato un bit per il segno, I bits per rappresentare la parte intera e D bits per rappresentare la parte decimale (ovviamente sarà N = I + D +1).

s	a_{I-1}	a_{I-2}	a_{I-3}	....	a_0	b_{-1}	b_{-2}	...	b_{-D}
Segno (1 bit)	Parte Intera (I bit)					Parte Frazionaria (D bit)			

$$Numero_{10} = (-1)^s \cdot \left(\sum_{i=0}^{I-1} a_i \cdot 2^i + \sum_{d=-1}^{-D} b_d \cdot 2^d\right)$$

Codifica dei Numeri Reali Virgola Fissa

Esempio. Convertire il numero binario 1010101 in virgola fissa

segno	a_2	a_1	a_0	b_{-1}	b_{-2}	b_{-3}
1	0	1	0.	1	0	1

$$N_{10} = (-1)^s \cdot (\sum_{i=0}^{I-1} a_i \cdot 2^i + \sum_{d=-1}^{-D} b_d \cdot 2^d)$$

- Segno: 1
- Parte Intera (I=3): $0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 2$
- Parte Frazionaria (D=3): $1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} = 0.5 + 0.125 = 0.625$

Quindi il numero decimale corrispondente a 1010101 è -2,625.

Codifica dei Numeri Reali Virgola Mobile

In un numero rappresentato in virgola mobile vengono stabiliti un certo numero di bit assegnati per codificare il segno (s), la mantissa (m) ed un certo numero di bit per codificare l'esponente (e).

s (1 bit)	e	m

E’ possibile in teoria avere un numero elevatissimo di codifiche in virgola mobile, cambiando:

- Numero bit
- Tipo di codifica binaria per mantissa ed esponente
- Formula di conversione decimale/binario e viceversa

Codifica dei Numeri Reali Virgola Mobile IEEE 754

- Nell'anno 1985 l'IEEE (The Institute of Electrical and Electronics Engineering) ha definito uno standard per la codifica dei numeri reali in virgola mobile: IEEE 754.
- Lo standard prevedeva inizialmente due codifiche a 32 (float) e a 64 bit (double). Esistono poi delle estensioni, ad esempio quella a 80 bits.
- La rappresentazione IEEE 754 prevede due diverse forme:
  - Normalizzata (default)
  - Denormalizzata (più precisa soprattutto per numeri piccoli, ma più complessa da realizzare)

Codifica dei Numeri Reali Virgola Mobile IEEE 754

- Nella codifica **float** a 32 bit, vengono assegnati:
  - 1 bit per il segno (il bit più significativo), s
  - 8 bit per l'esponente, e
  - 23 bit per la mantissa, m

- Nella codifica **double** a 64 bit, vengono assegnati:
  - 1 bit per il segno (il bit più significativo), s
  - 11 bit per l'esponente, e
  - 52 bit per la mantissa, m

Codifica dei Numeri Reali Virgola Mobile IEEE 754

Forma normalizzata (default)

s (1 bit)	e	m

$$N_{10} = (-1)^s \cdot f \cdot 2^{v-p}$$

- v=numero intero ottenuto da conversione binaria senza segno dei bit **e**
- $1 \leq v \leq max$
  - max=254 (8 bit-float), max=2046 (11 bit-double)
- p=polarizzazione (bias), valore fisso
  - p=127 (float), p=1023 (double)
- f = numero reale ottenuto da codifica in virgola fissa, **considerando m come parte frazionaria** e assumendo sempre che la **parte intera sia 1**

Codifica dei Numeri Reali Virgola Mobile IEEE 754

Forma normalizzata (default)

s (1 bit)	e	m

Valori Particolari

Zero	e=0	m=0	$N_{10} = 0$

Infinito	e=111111111..1	m=0	$N_{10} = \infty$

Codifica dei Numeri Reali Virgola Mobile IEEE 754

Esempi: Conversione da virgola mobile (forma normalizzata)

- 1 10000001 01000000000000000000000
  - Segno negativo –
  - v-p=$2^7+2^0-127=129-127=2$
  - f=$1+2^{-2}=1.25$
  - Numero=$-1.25*2^2=-5$
- 0 10000011 10011000000000000000000
  - Segno positivo +
  - v-p=$2^7+2^1+2^0-127=131-127=4$
  - f=$1+2^{-1}+2^{-4}+2^{-5}=1.59375$
  - Numero=$1.59375*2^4=25.5$

Codifica dei Numeri Reali Virgola Mobile IEEE 754

Estremi degli Intervalli Forma Normalizzata, float

- Più grande float normalizzato:

  X **11111110** 11111111111111111111

  - v-p=254-127=127
  - $f=1+2^{-1}+2^{-2}+2^{-3}+\ldots+2^{-23}\sim 2.0$
  - $\text{Numero}\sim\pm 2.0*2^{127}=\pm 2^{128}\sim 3.4*10^{38}$

- Più piccolo float normalizzato

  X **00000001** 00000000000000000000

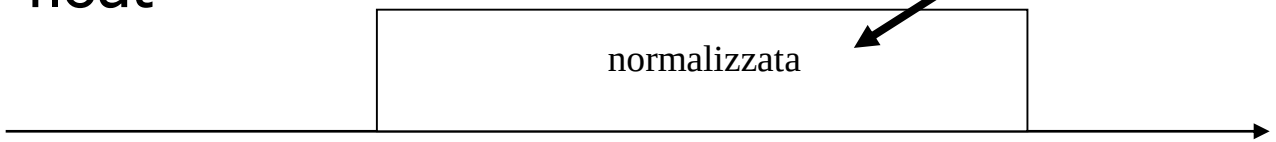
  - v-p=1-127=-126
  - f=1.0
  - $\text{Numero}=\pm 1.0*2^{-126}\sim 1.17*10^{-38}$

Codifica dei Numeri Reali Virgola Mobile IEEE 754

Valori Assoluti

$$N_{10} = (-1)^s \cdot f \cdot 2^{v-127}$$

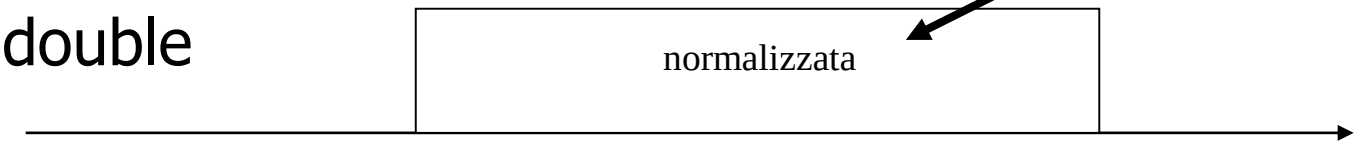
float



$\approx 1.17 \times 10^{-38}$ $\approx 3.4 \times 10^{38}$

$$N_{10} = (-1)^s \cdot f \cdot 2^{v-1023}$$

double



$1 \times 2^{-1022} \approx 2.225 \times 10^{-308}$ $\approx 1.7 \times 10^{308}$

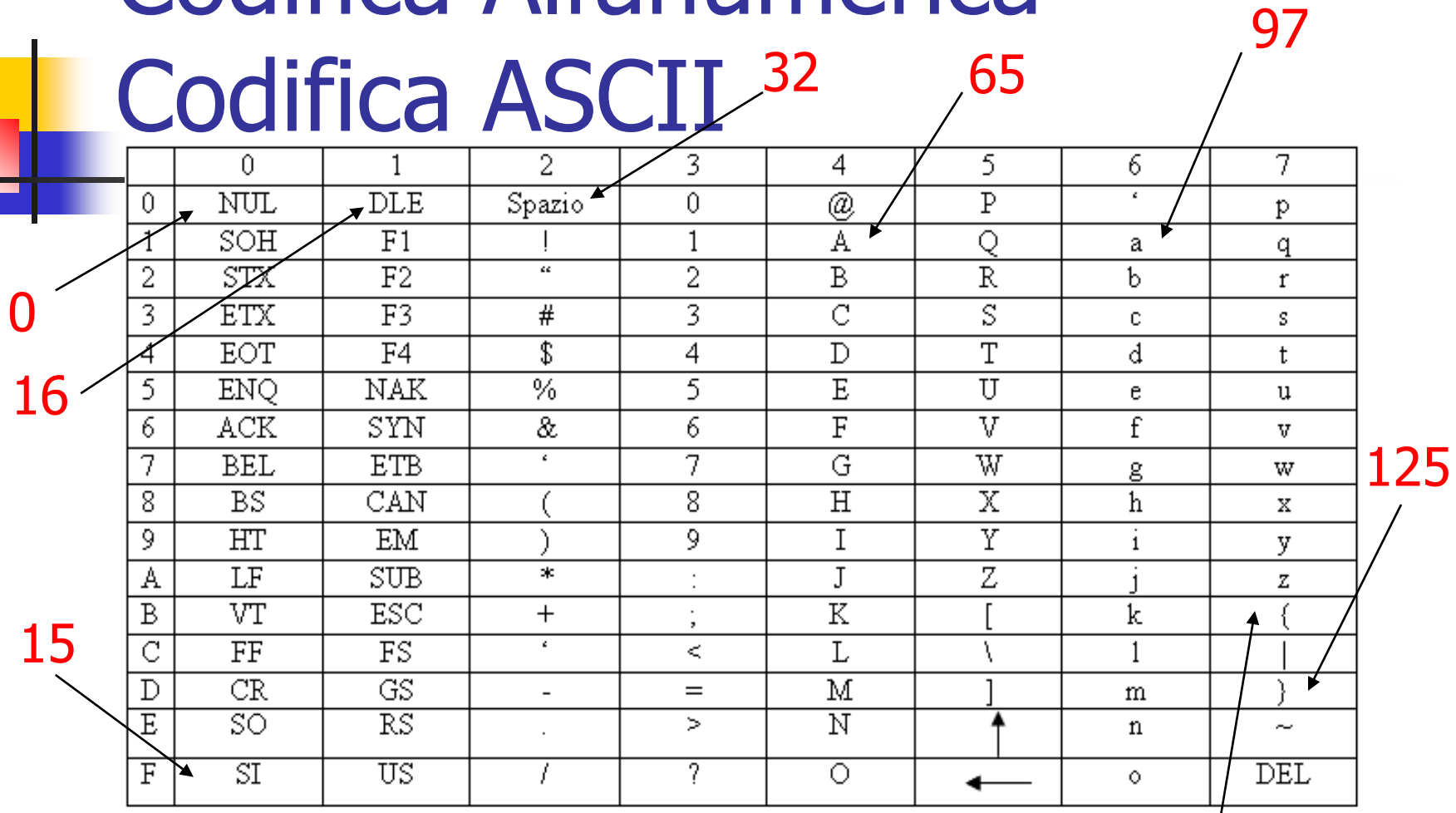
Codifica Alfanumerica

- L’insieme dei caratteri alfanumerici: lettere dell'alfabeto e dalle dieci cifre decimali. A questi vanno aggiunti diversi altri simboli come lo spazio, i segni di interpunzione, i simboli per indicare il passaggio alla riga o alla pagina successiva, ecc.
- Questo insieme di caratteri alfanumerici può essere facilmente rappresentato attribuendo in maniera univoca a ciascuno dei suoi elementi un numero intero (codice).
  - E‘ necessaria l'adozione di una comune e standard rappresentazione.
- Il numero totale delle lettere dell'alfabeto inglese sono 52, considerando anche quelle maiuscole. Aggiungendo le dieci cifre numeriche, una quarantina di simboli extra, arriviamo ad un totale di un centinaio di simboli da rappresentare.
  - Soli 7 bit sono sufficienti per rappresentare l'insieme dei caratteri alfanumerici (7 bit permettono di rappresentare 128 simboli diversi).

Codifica Alfanumerica
Codifica ASCII

- La codifica ASCII (che si pronuncia ASKI), prende il nome da American Standard Code for Information Interchange.
- Tale codifica si basa sull'utilizzo di 7 bit per un totale di 128 simboli rappresentabili.
- Da notare che i caratteri dell'alfabeto e le cifre numeriche successive hanno codice anch'esso successivo (ad esempio a ha codice 97, b codice 98, c codice 99, il numero 0 ha codice 48, il numero 1 codice 49, etc.)
- Tra le più utilizzate codifiche ASCII (entro i primi 128 simboli) vi sono:
  - ~ (tilde) codice 126
  - { codice 123
  - } codice 125
  - | codice 124

Codifica Alfanumerica
Codifica ASCII



	0	1	2	3	4	5	6	7
0	NUL	DLE	Spazio	0	@	P	‘	p
1	SOH	F1	!	1	A	Q	a	q
2	STX	F2	“	2	B	R	b	r
3	ETX	F3	#	3	C	S	c	s
4	EOT	F4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	‘	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	‘	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	↑	n	~
F	SI	US	/	?	O	←	o	DEL

Come si vede:

- I caratteri di controllo (non riproducibili) hanno i codici più bassi.
- Il blank (Spazio) è il primo dei caratteri riproducibili.
- Le maiuscole/minuscole sono ordinate (codice Progressivo).

Codifiche Alfanumeriche derivate dalla Codifica ASCII

- Esistono numerose estensioni della codifica ASCII.
- Tali estensioni derivano dalla necessità di codificare simboli legati a particolari lingue, e dal fatto che operando su 8 bit, la codifica ASCII consente l'utilizzo dell'ottavo bit, lasciando gli altri 7 inalterati.
- Tutte le estensioni della codifica ASCII non modificano tale codifica ma aggiungono semplicemente altri 128 simboli.
- Tra le estensioni STANDARD più diffuse vi è la ISO Latin1 (**Latino di Base**, caratteri per linguaggi europa occidentale)

Codifica Alfanumerica
Codifica Unicode

- La codifica Unicode supera i limiti della codifica ASCII e relativi derivati, in quanto estende il numero di simboli codificabili.
- Operava inizialmente su 2 byte (65.536 codifiche)
- E' stata estesa fino a **32 bit** e attualmente sono più di 100.000 simboli codificati con lo standard Unicode. Essi sono divisi in:
  - Script Moderni: Latino, Greco, Giapponese, Cinese, Koreano, etc.
  - Script Antichi: Sumero, Egiziano, etc.
  - Segni Speciali
- La codifica Unicode ingloba la codifica ISO Latin1.