Rappresentazione dell'Informazione

Emilio Di Giacomo e Walter Didimo

Rappresentaz. dell'informazione

- Abbiamo visto che l'informazione memorizzata ed elaborata da un calcolatore viene rappresentata come una sequenza di simboli che possono assumere *due soli valori*
  - convenzionalmente 0 e 1
- Vediamo quindi come, utilizzando un alfabeto di due soli simboli, è possibile rappresentare:
  - numeri interi
  - numeri "reali"
  - caratteri

Rappresentazione posizionale

- La modalità di scrittura dei numeri naturali che comunemente usiamo prevede di scrivere un numero nella forma

$$c_{k-1}c_{k-2}\ldots c_1c_0$$

  dove ognuno dei simboli c_i è una cifra da 0 a 9
- Il valore rappresentato dalla sequenza di cifre può essere calcolato come

$$c_{k-1}\cdot 10^{k-1}+c_{k-2}\cdot 10^{k-2}+\ldots c_1\cdot 10^1+c_0\cdot 10^0$$

Rappresentazione posizionale

- Esempio:

$$37579$$

- valore:

$$3 \cdot 10^4 + 7 \cdot 10^3 + 5 \cdot 10^2 + 7 \cdot 10^1 + 9 \cdot 10^0$$

Rappresentazione posizionale

- Quello descritto è un sistema di numerazione posizionale
- Il valore di ogni elemento della sequenza dipende dalla sua posizione:
  - ad esempio 37579 contiene due volte il simbolo 7 ma esso vale una volta 7000 e una volta 70
- In particolare quello visto è chiamato sistema di numerazione decimale (o in base 10), in quanto:
  - 10 è la base delle potenze che moltiplicano le cifre
  - 10 sono le cifre utilizzate (da 0 a 9)

Sistemi non posizionali

- Esistono sistemi di numerazione *non posizionali*
- Ad esempio il sistema di numerazione romano
- Nel numero romano XXVIII (cioè 28) compaiono due X ma esse hanno entrambe valore 10
- Anche le tre I hanno tutte valore 1

Rappresentazione posizionale

- È possibile utilizzare come base di un sistema di numerazione posizionale *qualunque numero naturale* $b \geq 2$
- Scelta una base b, un numero naturale viene rappresentato come

$$c_{k-1}c_{k-2}\ldots c_1c_0$$

  dove ognuno dei simboli c_i è una cifra da 0 a b-1
- Il valore rappresentato dalla sequenza di cifre può essere calcolato come

$$c_{k-1} \cdot b^{k-1} + c_{k-2} \cdot b^{k-2} + \ldots c_1 \cdot b^1 + c_0 \cdot b^0$$

Sistema binario

- Se scegliamo $b=2$ otteniamo il sistema di numerazione binario
- Le cifre possibili sono 0 e 1
- Esempio:

$$110111011$$

- valore:

$$1\cdot 2^8+1\cdot 2^7+0\cdot 2^6+1\cdot 2^5+1\cdot 2^4+1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0=$$

$$1\cdot 256+1\cdot 128+0\cdot 64+1\cdot 32+1\cdot 16+1\cdot 8+0\cdot 4+1\cdot 2+1\cdot 1=$$

$$443$$

Sistema binario

- Il sistema binario ci permette di rappresentare qualunque numero intero utilizzando i due soli simboli 0 e 1
- Può quindi essere utilizzato per rappresentare i numeri all’interno di un calcolatore

Sistemi ottale ed esadecimale

- Esistono altri due sistemi di numerazione molto utilizzati in Informatica
- Il sistema ottale in cui b=8
  - le cifre possibili sono da 0 a 7
- Il sistema esadecimale in cui b=16
  - si utilizzano le cifre da 0 a 9 (con ovvio valore)
  - e le cifre A, B, C, D, E e F che corrispondono ai valori 10, 11, 12, 13, 14 e 15

Sistemi ottale ed esadecimale

- Esempio (b=8):

  267

- valore:

  $2 \cdot 8^2 + 6 \cdot 8^1 + 7 \cdot 8^0 = 2 \cdot 64 + 6 \cdot 8 + 7 \cdot 1 = 183$

- Esempio (b=16):

  A3F

- valore:

  $A \cdot 16^2 + 3 \cdot 16^1 + F \cdot 16^0 = 10 \cdot 256 + 3 \cdot 16 + 15 \cdot 1 = 2623$

Notazione

- Nel seguito, quando necessario, indicheremo la base *b* di rappresentazione di un numero ponendo il simbolo $|_b$ a fianco del numero
- $267|_8$ indica la sequenza 267 in base 8, cioè il numero 183
- $267|_{16}$ indica invece la sequenza 267 in base 16, cioè il numero 615

Rappresentare *n* in base *b*

- Come possiamo ottenere la rappresentazione di un numero *n* in una certa base *b*?
- Esiste un procedimento semplice, che consiste nel dividere ripetutamente *n* per *b*
- Si divide *n* per *b*: quoziente n_1 e resto r_1
  - r_1 è la cifra più a destra della rappresentazione di *n* in base *b*
- Si divide n_1 per *b*: quoziente n_2 e resto r_2
  - r_2 è la prossima cifra della rappresentazione di *n* in base *b*
- Si continua finché il quoziente diviene 0

Rappresentare *n* in base *b*

- Rappresentiamo 116 in base 2

116/2 = 58 con il resto di 0

58/2 = 29 con il resto di 0

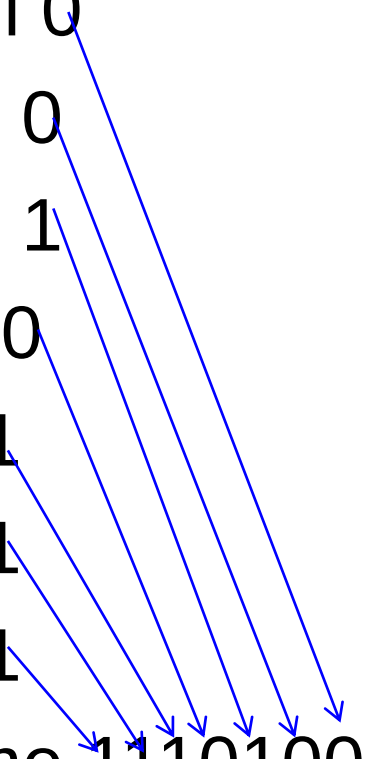
29/2 = 14 con il resto di 1

14/2 = 7 con il resto di 0

7/2 = 3 con il resto di 1

3/2 = 1 con il resto di 1

1/2 = 0 con il resto di 1



- 116 in base 2 è rappresentato come 1110100

Rappresentare *n* in base *b*

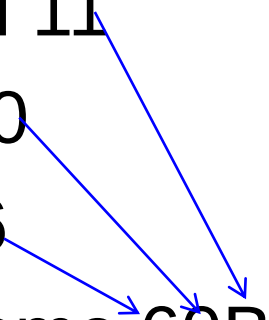
- Rappresentiamo 1547 in base 16

  1547/16 = 96 con il resto di 11

  96/16 = 6 con il resto di 0

  6/16 = 0 con il resto di 6

- 1547 in base 16 è rappresentato come 60B



Conversione di base

- Il procedimento visto può essere utilizzato anche per effettuare una conversione di base
- Sia data la rappresentazione di un numero *n* in una base *b*

$$c_{k-1}c_{k-2}\ldots c_1c_0|_b$$

- Per determinare la rappresentazione di *n* in un'altra base *b'* si può:
  - calcolare il valore *n* come $c_{k-1} \cdot b^{k-1}+\ldots c_1 \cdot b^1+c_0 \cdot b^0$
  - utilizzare il metodo delle divisioni successive per rappresentare *n* in base *b'*

Conversione di base: esempio

- Convertiamo in base 5 il numero $110111|_2$
- Calcoliamo il valore del numero:

$$1\cdot2^5+1\cdot2^4+0\cdot2^3+1\cdot2^2+1\cdot2^1+1\cdot2^0=55$$

- Rappresentiamo 55 in base 5

55/5 = 11 con il resto di 0

11/5 = 2 con il resto di 1

2/5 = 0 con il resto di 2

- Quindi $110111|_2 = 210|_5$

Ulteriori considerazioni

- Possiamo utilizzare basi diverse dalla base 10 anche per rappresentare i numeri reali
- Usiamo la virgola per separare la parte intera dalla parte frazionaria
- Le cifre a destra della virgola vanno moltiplicate per le potenze negative della base

Ulteriori considerazioni

- Esempi:

  $1010{,}0101|_2 =$

  $1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} =$

  $8 + 2 + 1/4 + 1/16 = 10{,}3125$

  $A4CF{,}3B|_{16} =$

  $(10 \cdot 16^3 + 4 \cdot 16^2 + 12 \cdot 16^1 + 15 \cdot 16^0 + 3 \cdot 16^{-1} + B \cdot 16^{-2}) =$

  $42191{,}23046875$

Ulteriori considerazioni

- Anche l'aritmetica può essere eseguita nella maniera usuale

```
1     1          Riporti
  1 0 0 1 0 +    (18)
  1 1 0 1 1 =    (27)
-----------------
1 0 1 1 0 1      (45)
```

```
      1 0 1 0 ×    (10)
        1 1 0 =    (6)
-------------------
    0 0 0 0
  1 0 1 0
1 0 1 0
-------------------
1 1 1 1 0 0        (60)
```

Rappresentazione di interi nel calcolatore

Numeri interi nel calcolatore

- La rappresentazione dei numeri interi all'interno del calcolatore ci impone:
  - di usare le cifre 0 e 1 (rappresentazione binaria)
  - di usare un *numero finito di cifre*: i numeri devono infatti essere memorizzati nei registri o nelle celle di memoria che hanno una dimensione fissa (es. 32 bit)
  - trovare un modo di rappresentare i *numeri negativi*: non possiamo usare il segno meno

Numero di bit finito

- Poiché dobbiamo utilizzare un numero finito di bit, i numeri che possiamo utilizzare sono in *numero finito*:
  - alcuni numeri *non sono rappresentabili*
- Se utilizziamo *k* bit per rappresentare gli interi, possiamo rappresentare 2^k diversi valori
  - se ci limitiamo ai numeri naturali l'intervallo rappresentabile è $[0,2^k-1]$
  - ad esempio con 3 bit possiamo rappresentare l'intervallo [0,7]

Overflow

- Poiché l'intervallo rappresentabile è finito, il risultato di un'operazione aritmetica può cadere al di fuori di tale intervallo
- Il risultato non sarà quindi rappresentabile
- Si parla di errore di overflow (o di trabocco)

Overflow

- Ad esempio:
  - se utilizziamo 3 bit e sommiamo $101|_2$ (cioè 5) e $100|_2$ (cioè 4) il risultato (cioè 9) richiede 4 bit per essere rappresentato
- La condizione di overflow è segnalata dalla presenza di un riporto sulla cifra più significativa

1		*Riporti*
	1 0 1 +	*(5)*
	1 0 0 =	*(4)*
1	0 0 1	*(9)*

Rappres. dei numeri negativi

- Poiché gli unici simboli ammessi sono 0 e 1, non possiamo utilizzare il segno – per indicare i numeri negativi
- Dobbiamo utilizzare opportune *codifiche*
- Esistono diverse codifiche proposte per la rappresentazione degli interi (positivi e negativi):
  - modulo e segno
  - complemento a 1
  - complemento a 2

Modulo e segno

- Il bit più significativo viene interpretato come bit di segno
  - 0 corrisponde a +, 1 corrisponde a –
- I restanti bit codificano il modulo del numero secondo la codifica binaria “standard”
- Esempio (4 bit):
  - 0110 equivale a 6, 1110 equivale a -6
- Intervallo rappresentabile (*k* bit): $[-2^{k-1}+1, 2^{k-1}-1]$
- Due rappresentazioni per lo zero: 0000 e 1000

Complemento a uno

- Un numero positivo viene rappresentato in binario (il bit più a sinistra deve essere 0)
- Per rappresentare un numero negativo *x* si rappresenta prima in binario –*x* (con il bit più a sinistra a 0) e poi si invertono tutti i bit
- Esempio (4 bit):
  - 0110 equivale a 6, 1001 equivale a -6
- Intervallo rappresentabile (*k* bit): $[-2^{k-1}+1, 2^{k-1}-1]$
- Due rappresentazioni per lo zero: 0000 e 1111

Complemento a uno

- Anche in questo caso il bit più significativo è pari a 0 per i numeri positivi e pari ad 1 per i negativi
- Data la rappresentazione in complemento a uno $c_{k-1}c_{k-2}...c_1c_0$ di un numero, è anche possibile calcolare il valore del numero come:

$$c_{k-1} \cdot(-2^{k-1}+1)+c_{k-2}\cdot 2^{k-2}+...c_1\cdot 2^1+c_0\cdot 2^0$$

Complemento a due

- Un numero positivo viene rappresentato in binario (il bit più a sinistra deve essere 0)
- Per rappresentare un numero negativo si somma 1 alla sua rappresentazione in complemento a uno
- Esempio (4 bit):
  - 0110 è pari a 6, 1010 è pari a -6:
    - -6 in complemento a 1 è 1001
    - -6 in complemento a 2 è dunque: 1001+1=1010
- Intervallo rappresentabile (k bit): $[-2^{k-1},2^{k-1}-1]$
- Una sola rappresentazione per lo zero: 0000

Complemento a due

- Anche in questo caso il bit più significativo è pari a 0 per i numeri positivi e pari ad 1 per i negativi
- Data la rappresentazione in complemento a due $c_{k-1}c_{k-2}...c_1c_0$ di un numero, è anche possibile calcolare il valore del numero come:

$$c_{k-1} \cdot(-2^{k-1})+c_{k-2}\cdot 2^{k-2}+...c_1\cdot 2^1+c_0\cdot 2^0$$

Confronto (k=4)

Valore	Modulo e segno	Complemento a 1	Complemento a 2	Eccesso 2^{k-1}
-8	-	-	1000	0000
-7	1111	1000	1001	0001
-6	1101	1001	1010	0010
-5	1101	1010	1011	0011
-4	1100	1011	1100	0100
-3	1011	1100	1101	0101
-2	1010	1101	1110	0110
-1	1001	1110	1111	0111
0	0000/1000	0000/1111	0000	1000
1	0001	0001	0001	1001
2	0010	0010	0010	1010
3	0011	0011	0011	1011
4	0100	0100	0100	1100
5	0101	0101	0101	1101
6	0110	0110	0110	1110
7	0111	0111	0111	1111

Quale usiamo?

- La rappresentazione utilizzata più diffusamente è la rappresentazione in complemento a due
- Essa permette di eseguire in maniera più semplice le operazioni aritmetiche
  - si possono realizzare circuiti di calcolo più semplici all'interno del calcolatore
- Ad esempio, la somma può essere eseguita allo stesso modo qualunque sia il segno degli operandi

Somma in complemento a 2

- Esempi:

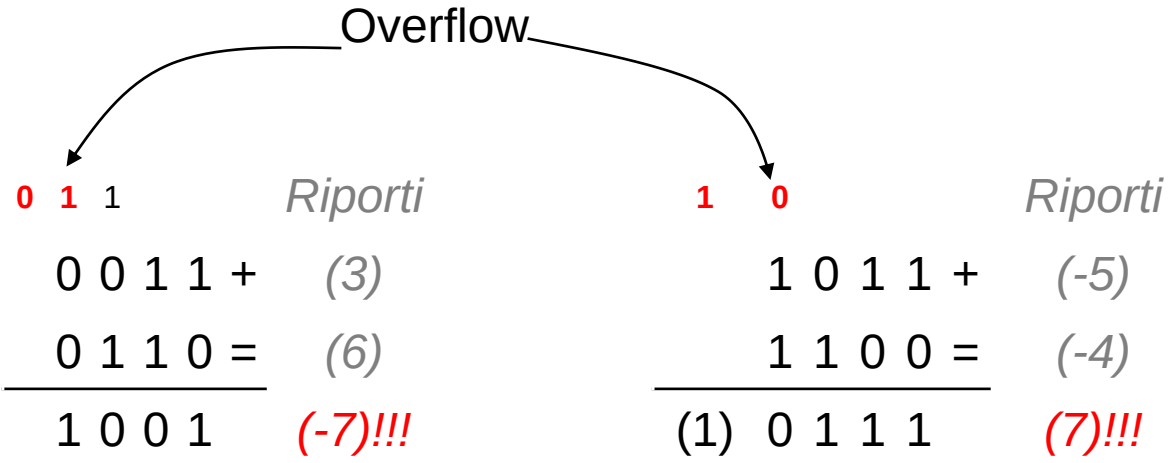
	Riporti			Riporti			Riporti
	1			1 1 1			1 1
0 0 1 0 +	(2)		1 0 1 1 +	(-5)		1 0 1 1 +	(-5)
0 0 1 1 =	(3)	Ignorato	1 1 1 0 =	(-2)		0 0 1 1 =	(3)
0 1 0 1	(5)		(1) 1 0 0 1	(-7)		1 1 1 0	(-2)

- Nota: in questo caso un eventuale bit di overflow va ignorato (non c'è un effettivo errore di overflow)

Complemento a 2 e overflow

- L'errore di overflow si ha quando i due operandi hanno lo stesso segno ma il risultato ha segno opposto
  - in questo caso il riporto sulla penultima posizione è diverso dal riporto sull'ultima posizione

- Esempi:



Rappresentazione dei numeri reali

Rappresentazione dei numeri reali

- Per i numeri reali si utilizza la rappresentazione in virgola mobile (floating point)
- Tale rappresentazione usa la stessa idea della notazione scientifica
- Utilizzando la notazione scientifica, un numero viene rappresentato nella forma $m \cdot 10^e$
  - m viene detta mantissa
  - e viene detto esponente
- Esempio:
  - la notazione $1{,}34 \cdot 10^5$ rappresenta il numero 134000

Notazione scientifica

- Utilizzando la notazione scientifica, lo stesso numero può essere rappresentato da diverse coppie *mantissa-esponente*
- Esempio:
  - $1{,}34 \cdot 10^5$, $134 \cdot 10^3$, $0{,}0134 \cdot 10^7$ sono rappresentazioni diverse del numero 134000
- Per avere una rappresentazione univoca di solito si utilizza una forma *normalizzata* per la mantissa:
  - una sola cifra diversa da zero a sinistra della virgola
  - $1{,}34 \cdot 10^5$ è la rappresentazione normalizzata di 134000

Notazione scientifica

- Il vantaggio principale della notazione scientifica è che ci permette di rappresentare numeri molto grandi o molto piccoli in maniera compatta
  - Distanza dal bordo dell'universo osservabile: $4{,}6 \cdot 10^{26}\ m$
  - Massa di un protone: $1{,}6726231 \cdot 10^{-27}\ kg$

Rappresentazione in virgola mobile

- La rappresentazione in virgola mobile segue la stessa logica della notazione scientifica
- Ogni numero viene rappresentato mediante:
  - un bit di segno *s* (0 per + e 1 per -)
  - una mantissa *m*
  - un esponente *e*
- Il valore rappresentato è $(-1)^{s} \cdot m \cdot 2^{e}$
- *m* ed *e* sono rappresentati tramite una qualche codifica binaria con un numero fisso di bit
  - vedremo i dettagli più avanti

Virgola mobile: limiti

- Il fatto di utilizzare un numero finito di bit per mantissa ed esponente causa una serie di limitazioni nella rappresentazione dei numeri reali
- Per descriverle facciamo riferimento alla notazione scientifica in base dieci
  - assumiamo di usare 3 cifre per la mantissa normalizzata e 2 per l'esponente

Virgola mobile: limiti

- Il numero positivo più grande che possiamo rappresentare è $9{,}99 \cdot 10^{99}$
  - se il risultato di un'operazione è maggiore di $9{,}99 \cdot 10^{99}$ si ha un errore di overflow
- Il numero positivo più piccolo che possiamo rappresentare è $1{,}00 \cdot 10^{-99}$
  - se il risultato di un'operazione è positivo e minore di $1{,}00 \cdot 10^{-99}$ si ha un errore di underflow (di solito arrotondato 0)
- Considerazione analoghe valgono per i numeri negativi

Virgola mobile: limiti

- Consideriamo un valore, per esempio $2{,}13 \cdot 10^{15}$
- Il valore più vicino a $2{,}13 \cdot 10^{15}$ che possiamo rappresentare è $2{,}14 \cdot 10^{15}$
- Tutti i numeri compresi tra $2{,}13 \cdot 10^{15}$ e $2{,}14 \cdot 10^{15}$ (*che sono infiniti*) non possono essere rappresentati
  - se un risultato cade tra $2{,}13 \cdot 10^{15}$ e $2{,}14 \cdot 10^{15}$ viene approssimato al più vicino dei due
  - si ha un errore di approssimazione

Lo standard IEEE 754

- La modalità di rappresentazione in virgola mobile comunemente adottata è quella definita nello Standard IEEE 754
- Tale standard definisce diversi possibili formati
- Noi vedremo:
  - il formato a precisione singola (32 bit)
  - il formato a precisione doppia (64 bit)

Lo standard IEEE 754

- Entrambi i formati utilizzano un bit di segno
- Per l'esponente si usano:
  - 8 bit in precisione singola
  - 11 bit in precisione doppia
- Per la mantissa si usano:
  - 23 bit in precisione singola
  - 52 bit in precisione doppia

Lo standard IEEE 754

- Zero:
  - esistono due rappresentazioni corrispondenti ai due diversi valori del bit di segno
- Infinito:
  - usato per arrotondare in certi casi un risultato che abbia causato un errore di overflow
  - può essere usato come operando
- NaN (Not a Number):
  - usato per rappresentare il risultato di operazioni che danno luogo a forme indeterminate come 0/0 o ∞/∞

Lo standard IEEE 754

- Valori massimi e minimi (positivi) rappresentabili

		Esponente	Mantissa	Valore
Precisione singola	Min. num. denormalizzato	-126	2^{-23}	$\cong 1{,}4 \cdot 10^{-45}$
	Max. num. denormalizzato	-126	$(1-2^{-23})$	$\cong 1{,}18 \cdot 10^{-38}$
	Min. num. normalizzato	-126	1	$\cong 1{,}18 \cdot 10^{-38}$
	Max. num. normalizzato	127	$(2-2^{-23})$	$\cong 3{,}40 \cdot 10^{38}$
Precisione doppia	Min. num. denormalizzato	-1022	2^{-52}	$\cong 4{,}9 \cdot 10^{-324}$
	Max. num. denormalizzato	-1022	$(1-2^{-52})$	$\cong 2{,}23 \cdot 10^{-308}$
	Min. num. normalizzato	-1022	1	$\cong 2{,}23 \cdot 10^{-308}$
	Max. num. normalizzato	1023	$(2-2^{-52})$	$\cong 1{,}80 \cdot 10^{308}$

Rappresentazione di caratteri

Rappresentazione di caratteri

- Per rappresentare i caratteri si associa ad ogni carattere un codice numerico
- Il codice viene poi codificato in binario
- Immaginiamo di voler codificare un alfabeto dei soli caratteri A, B, C, D, E e F

Carattere	Codice numerico	Rappresentazione binaria
A	1	001
B	2	010
C	3	011
D	4	100
E	5	101
F	6	110

Il codice ASCII

- Uno dei primi standard definiti e utilizzati a livello internazionale è il codice ASCII (*American Standard Code for Information Interchange*)
  - Definito dall'American Standard Association nel 1963
  - successivamente aggiornato nel 1967 e nel 1986
- Utilizza 7 bit per codificare i caratteri
- 128 caratteri rappresentabili:
  - 33 caratteri di controllo
  - 95 caratteri stampabili

Il codice ASCII

Caratteri stampabili del codice ASCII

Binario	Dec.	Car.	Binario	Dec.	Car.	Binario	Dec.	Car.
010 0000	32		100 0000	64	@	110 0000	96	‘
010 0001	33	!	100 0001	65	A	110 0001	97	a
010 0010	34	"	100 0010	66	B	110 0010	98	b
010 0011	35	#	100 0011	67	C	110 0011	99	c
010 0100	36	$	100 0100	68	D	110 0100	100	d
010 0101	37	%	100 0101	69	E	110 0101	101	e
010 0110	38	&	100 0110	70	F	110 0110	102	f
010 0111	39	’	100 0111	71	G	110 0111	103	g
010 1000	40	(	100 1000	72	H	110 1000	104	h
010 1001	41	)	100 1001	73	I	110 1001	105	i
010 1010	42	*	100 1010	74	J	110 1010	106	j
010 1011	43	+	100 1011	75	K	110 1011	107	k
010 1100	44	,	100 1100	76	L	110 1100	108	l
010 1101	45	-	100 1101	77	M	110 1101	109	m
010 1110	46	.	100 1110	78	N	110 1110	110	n
010 1111	47	/	100 1111	79	O	110 1111	111	o
011 0000	48	0	101 0000	80	P	111 0000	112	p
011 0001	49	1	101 0001	81	Q	111 0001	113	q
011 0010	50	2	101 0010	82	R	111 0010	114	r
011 0011	51	3	101 0011	83	S	111 0011	115	s
011 0100	52	4	101 0100	84	T	111 0100	116	t
011 0101	53	5	101 0101	85	U	111 0101	117	u
011 0110	54	6	101 0110	86	V	111 0110	118	v
011 0111	55	7	101 0111	87	W	111 0111	119	w
011 1000	56	8	101 1000	88	X	111 1000	120	x
011 1001	57	9	101 1001	89	Y	111 1001	121	y
011 1010	58	:	101 1010	90	Z	111 1010	122	z
011 1011	59	;	101 1011	91	[	111 1011	123	{
011 1100	60	<	101 1100	92	\	111 1100	124	|
011 1101	61	=	101 1101	93	]	111 1101	125	}
011 1110	62	>	101 1110	94	^	111 1110	126	~
011 1111	63	?	101 1111	95	_			

Lettere maiuscole

Lettere minuscole

Cifre

Oltre il codice ASCII

- Il codice ASCII non permette di rappresentare lettere con accenti o altri segni diacritici (come è, é, ü, ç, ecc.)
- Tali lettere non sono usate in inglese ma sono utilizzate in molte lingue europee
- Per questo motivo sono nate diverse estensioni dell'ASCII per aggiungere il supporto per tali caratteri
- Tali estensioni di solito utilizzano 8 bit, contengono l'ASCII e utilizzano i codici da 128 a 255 per i caratteri aggiuntivi

Oltre il codice ASCII

- Tra le estensioni dell'ASCII rientrano gli standard ISO 8859 che sono una famiglia di standard a 8 bit, ognuno dei quali estende l'ASCII originale
- Fa parte di questa famiglia lo standard ISO 8859-1 (ISO Latin-1) che estende l'ASCII con i caratteri delle lingue dell'Europa occidentale

Oltre ISO 8859

- Sebbene la famiglia ISO 8859 estende l'insieme dei caratteri rappresentabili essa non risolve tutti i problemi
- Le varie estensioni coincidono sui caratteri ASCII ma non su quelli estesi
  - lo scambio di dati tra sistemi con codifiche diverse è problematica
- Il numero di caratteri disponibili negli standard ISO 8859 non è sufficiente a supportare le lingue asiatiche (cinese, giapponese, ecc.)

Unicode

- Per superare i problemi visti, negli anni ’90 è stato creato il consorzio Unicode per la creazione di un sistema di codifica universale
- Il risultato è il codice Unicode, oggi ampiamente utilizzato come sistema di codifca di molti linguaggi (XML, Java, ecc.) e sistemi operativi
- Originariamente progettato per utilizzare 16 bit è stato poi esteso ed oggi prevede l’utilizzo di 1112064 codici

Unicode

- Unicode è pensato per supportare tutte le lingue esistenti, quelle morte, i simboli matematici, musicali, ecc.
- Il codice è ancora in evoluzione e alcuni codici non sono ancora assegnati
- I codici Unicode vengono di solito indicati con la notazione U+ seguita da 4 o 6 cifre esadecimali che rappresentano il codice:
  - ad es. U+0041 corrisponde al valore decimale 65 che è il codice associato al carattere A.

Unicode

- L’ampliamento dell’insieme di caratteri pone un problema
- Il numero di bit necessari a rappresentare un carattere è maggiore
  - i dati testuali aumentano di dimensione
- Per rappresentare 1112064 valori diversi sono necessari 3 byte:
  - i dati testuali assumerebbero dimensioni 3 volte maggiore che con i codici ISO 8859

Unicode e codifica di caratteri

- Per evitare l'aumento di dimensione dei dati testuali, in Unicode si usano diverse codifiche di carattere
- Si distingue tra i codici assegnati ai caratteri (code point) e il modo con cui tali codici vengono rappresentati in binario (codifica dei caratteri)

Unicode e codifica di caratteri

- Ad esempio, il code point del carattere A è 65
- La codifica di caratteri definisce come rappresentare in binario tale valore
- il modo più semplice consiste nel codificare 65 in binario, cioè come 1000001 (o 01000001 se usiamo 8 bit):
  - ciò è quanto succede in ASCII e ISO 8859
- Unicode prevede invece codifiche diverse, tra cui UTF-8, UTF-16, UTF-32

Un esempio di codifica: UTF-8

- Con UTF-8 si usano da 1 a 4 byte con le seguenti regole:
  - code point da 0 a 127, un byte nella forma 0*xxxxxxx*
  - code point da 128 a 2047, due byte nella forma 110*xxxxx* 10*xxxxxx*
  - code point da 2048 a 65535, tre byte nella forma 1110*xxxx* 10*xxxxxx* 10*xxxxxx*
  - code point oltre 65535, quattro byte nella forma 11110*xxx* 10*xxxxxx* 10*xxxxxx* 10*xxxxxx*
- In tutti i casi le *x* indicano i bit del code point

Un esempio di codifica: UTF-8

- Ad esempio:
  - il carattere A ha code point 65 che in binario è 1000001
  - poiché 65 cade tra 0 e 127 si usa il formato 0*xxxxxxx*
  - quindi la codifica UTF-8 di A è 01000001
  - il carattere € ha code point 8364 che in binario è 0010000010101100
  - poiché 8364 è compreso tra 2048 e 65535 si usa il formato 1110*xxxx* 10*xxxxxx* 10*xxxxxx*
  - quindi la codifica UTF-8 di € è 11100010 10000010 10101100

UTF-8: commenti

- Per i caratteri rappresentabili con il codice ASCII, la codifica UTF-8 coincide con la codifica ASCII
  - compatibilità con i vecchi sistemi che utilizzano ASCII
- Sebbene UTF-8 possa usare fino a 4 byte, la maggior parte dei caratteri sono codificati con uno o due byte
  - si contiene l'aumento di dimensione dei dati testuali