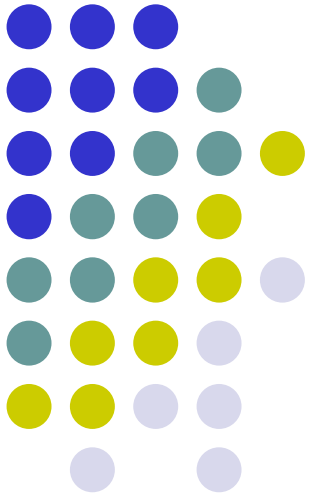
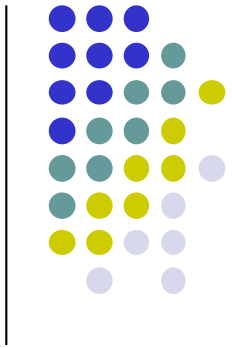
Rappresentazione dell'informazione

- Codifica binaria
- Sistemi di numerazione
- Codifica di caratteri, numeri, …





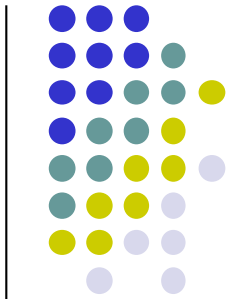
Codifica binaria dell'informazione: modalità di rappresentazione dei dati (numeri, caratteri, immagini, …) in un elaboratore

- Bit:
  - da binary digit
  - può avere soltanto valore 0 o 1
  - è l'unità elementare di informazione memorizzabile in un elaboratore
  - il suo valore corrisponde ad un possibile stato di un dispositivo bistabile (es., interruttore aperto o chiuso, cerchietto di semiconduttore magnetizzato o smagnetizzato, presenza o assenza di corrente o tensione, ….

- Byte = 8 bit.
- Kbyte = 2^{10} byte = 1024 byte
- Mbyte = $2^{10} \times 2^{10} = 2^{20}$ byte
- Gbyte = 2^{30} byte
- Tbyte = 2^{40} byte

- Es: una RAM di 512 Mbyte contiene 512×2^{20} byte o $512 \times 2^{20} \times 8$ bit

- Poiché $2^{10} \approx 1000$, a volte si approssimano K con mille, M con milione, G con miliardo, …



- Internamente ad un elaboratore tutto viene codificato con sequenze di bit in modo trasparente all'utente, poiché il procedimento inverso di presentazione delle informazioni secondo i nostri formalismi e percezioni è effettuato in modo automatico dai dispositivi di Input/Output

- Un bit permette di distinguere tra loro solo due elementi o possibilità, rappresentate o codificate rispettivamente dai valori *0* ed *1*

- Per codificare un numero superiore di elementi, è necessario utilizzare sequenze di bit, il cui numero aumenta all'aumentare della loro lunghezza

- Il numero di sequenze di lunghezza *k* è dato dal numero di possibili configurazioni ottenibili assegnando valori nell'ordine ai *k* bit, ossia 2^k

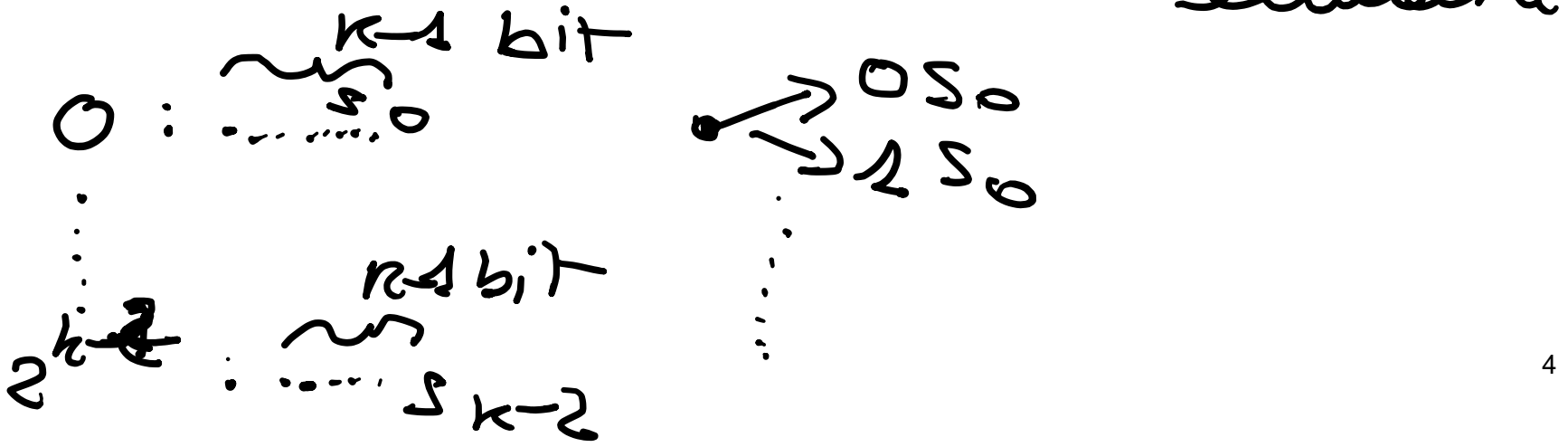
2^k elementi $\Rightarrow$ k bit

$k=1 \rightarrow \{0,1\}$

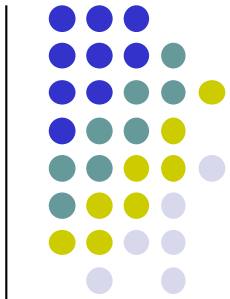
$k=2 \rightarrow \{00,01,10,11\}$

$\vdots$

s ha vero $k-1 \Rightarrow 2^{k-1}$ elementi

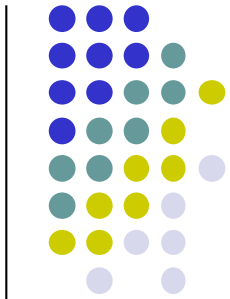


$$k-1 \Rightarrow 2^{k-1}$$



$$k \Rightarrow 2^{k-1} \cdot 2^{+1} = 2^{k}$$

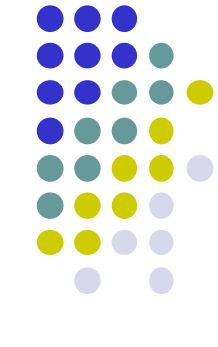
Codifica binaria



- Dato un insieme *U* di *n* elementi ed un numero intero *k* fissato, una codifica binaria di *U* a *k* bit consiste nell'assegnamento di sequenze di *k* bit agli elementi di *U* in modo univoco, ossia in modo tale che ad ogni elemento un'unica sequenza e ad ogni sequenza al più un elemento
- Più formalmente una codifica è una funzione iniettiva $c:U \rightarrow \{0,1\}^k$

Esempio

- *U={Luca, Marco, Alessio}*
- Codifica di *U* a *2* bit: *c(Luca)=00, c(Marco)=01* e *c(Alessio)=10*



- Domanda: quanto deve essere grande *k* per codificare un insieme *U* di *n* elementi?

- Con *k=1* si hanno soltanto *2* sequenze diverse, ossia *0* e *1*, per cui si possono codificare insiemi *U* con $n \leq 2$, ossia contenenti al più *2* elementi.
- Con *k=2* si hanno soltanto *4* sequenze diverse, ossia *00, 01, 10* e *11*, per cui si possono codificare insiemi *U* con $n \leq 4$
- Con *k=3* si hanno soltanto *8* sequenze diverse, per cui $n \leq 8$
- …
- Per un generico *k* deve essere $n \leq 2^k$, *ossia* $k \geq \log_2 n$

- In generale si cerca di scegliere *k* più piccolo possibile, in modo da risparmiare memoria
- Poiché *k* è un numero intero, il più piccolo valore possibile è $\boldsymbol{k} = \lceil \log_2 n \rceil$

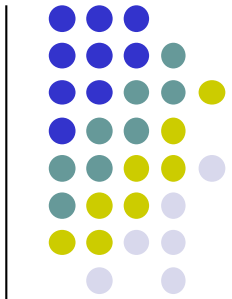
Riepilogando ...

n	*min k*
2	*1*
3	*2*
4	*2*
5	*3*
6	*3*
7	*3*
8	*3*
9	*4*
…	*…*

n	*min k*
16	*4*
17	*5*
…	*…*
32	*5*
33	*6*
…	*…*
64	*6*
65	*7*
…	*..*

n	*min k*
128	*7*
129	*8*
…	*…*
256	*8*
257	*9*
…	*…*
512	*9*
513	*10*
…	*…*

Codifica caratteri



Includono

- caratteri alfanumerici (lettere e cifre)
- simboli (punteggiatura, operatori aritmetici, …)
- caratteri di controllo (per formattazione, stampe, codici di trasmissione,…)

Alcuni codici noti: ASCII, EBCDIC, FIELDDATA, …

ASCII (American Standard Code for Information Interchange):

- 7 bit per carattere
- 1 bit per controllo errore (bit di parità, assegnato in modo da avere un numero pari di bit uguali ad 1 in tutti il byte)
- 128 caratteri rappresentati

Extended ASCII:

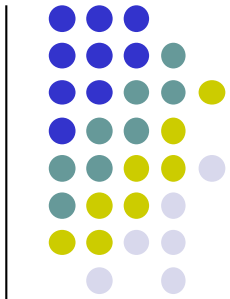
- 8 bit per carattere
- 256 caratteri rappresentati

(*Bit più significativi*) 4

				0	0	0	0	0	0	0	0
				0	0	0	0	1	1	1	1
				0	0	1	1	0	0	1	1
				0	1	0	1	0	1	0	1
1											
0	0	0	0	NUL	DLE	Spaz	0	@	P	`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NACK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

Tabella 2.6 La codifica ASCII standard. È immediato esprimere in forma esadecimale la codifica corrispondente a un dato carattere. Ad esempio, la codifica del carattere di spazio ("Spaz") è 20, quella del carattere "A" è 41, quella del carattere "9" è 39, mentre quella del "LF" (*Line Feed*) è A.

Codifica numeri

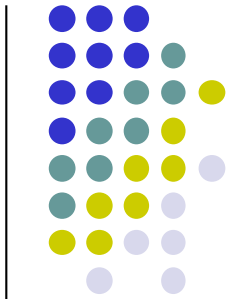


Codifiche per

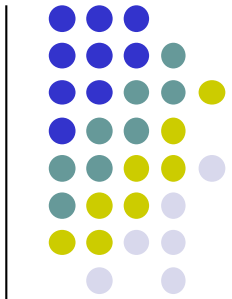
- naturali (interi non negativi)
- interi (anche negativi)
- frazionari (reali compresi tra *0* e *1*)
- reali

Le codifiche dei numeri non sono arbitrarie, ma progettate per facilitare lo svolgimento delle operazioni aritmetiche e garantire diversi livelli di precisione

Sistemi di numerazione



- Domanda: che cos'è un numero?
- Risposta: concetto astratto che corrisponde alla descrizione quantitativa degli oggetti contenuti in un determinato insieme
- Sistema di numerazione: insieme di simboli e regole atti a rappresentare i numeri
- Es: il numero di dita di una mano puo' essere rappresentato con I simboli 5, V (sistema di numerazione romano), …



Sistema di numerazione posizionale in base *10*:

- utilizza *10* cifre (*0, …, 9*)
- il valore di ogni cifra dipende dalla posizione all'interno del numero

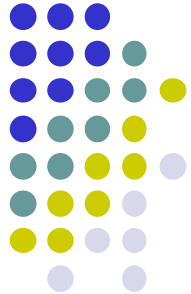
  Es. *357=7+5·10+3·100*
- in generale

$$b_{n-1}b_{n-2}\cdots b_1 b_0 =$$

$$= b_0 \cdot 10^0 + b_1 \cdot 10^1 + \cdots + b_{n-2} \cdot 10^{n-2} + b_{n-1} \cdot 10^{n-1} =$$

$$= \sum_{i=0}^{n-1} b_i \cdot 10^i$$

È possibile estendere a qualsiasi base *B>0*



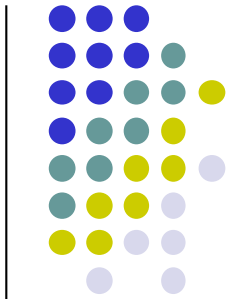
Sistema di numerazione posizionale in base *B*:

- utilizza *B* cifre
- il valore di ogni cifra dipende dalla posizione all'interno del numero:

$$(b_{n-1}b_{n-2}\cdots b_1 b_0)_B =$$

$$= b_0 \cdot B^0 + b_1 \cdot B^1 + \cdots + b_{n-2} \cdot B^{n-2} + b_{n-1} \cdot B^{n-1} =$$

$$= \sum_{i=0}^{n-1} b_i \cdot B^i$$

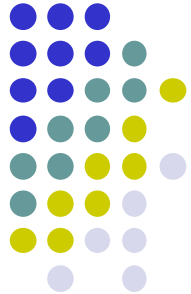


Quando la base *B* è maggiore di *10*, per indicare le cifre superiori a *9* per convenzione si utilizzano le lettere maiuscole dell'alfabeto

Esempio: *B=16* (sistema esadecimale)

Cifra in base *16*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
Valore in base *10*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*

Esempi di numerazioni



Base 10	Base 2	Base 3	Base 4	Base 5	Base 8	Base 16
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	10	2	2	2	2	2
3	11	10	3	3	3	3
4	100	11	10	4	4	4
5	101	12	11	10	5	5
6	110	20	12	11	6	6
7	111	21	13	12	7	7
8	1000	22	20	13	10	8
9	1001	100	21	14	11	9
10	1010	101	22	20	12	A
11	1011	102	23	21	13	B
12	1100	110	30	22	14	C
13	1101	111	31	23	15	D
14	1110	112	32	24	16	E
15	1111	120	33	30	17	F
16	10000	121	100	31	20	10

Tabella 2.1 I primi 17 numeri nelle basi 10, 2, 3, 5, 8 e 16.

$B = 10$

$123 = 3 \cdot 10^0 + 2 \cdot 10^1 + 1 \cdot 10^2 =$

$= 3 + 20 + 100 = \underline{123}$

$B = 2$

$1011 = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 =$

$= 1 + 2 + 8 = \underline{11}$

$B = 16$ sist.

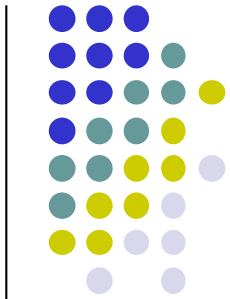
$ACA = 7 \cdot 16^0 + C \cdot 16^1 + A \cdot 16^2 =$ sist. ?

$= 7 \cdot 16^0 + 12 \cdot 16^1 + 10 \cdot 16^2 =$

$= 7 + (12 \cdot 16) + (10 \cdot 16 \cdot 16) =$

= fare i calcoli!

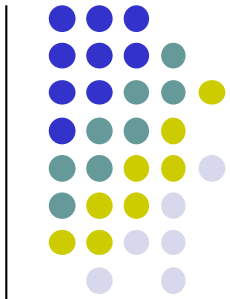
Conversioni di base



- Come convertire un numero in base *B* nel suo equivalente in una base *B*′≠ *B*?

- Vedremo ora due regole generali, la cui distinzione fondamentale consiste nel fatto che effettuano le operazioni aritmetiche coinvolte rispettivamente nella base partenza *B* e nella base di arrivo *B*′

Regola 1



- Svolge le operazioni nella base di arrivo *B'*, per cui è molto adatta al caso in cui *B'=10*
  - solitamente, si usa per convertire da qualsiasi base alla base 10

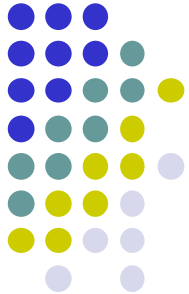
- Consiste nell'applicare in modo diretto la sommatoria

$$(b_{n-1}b_{n-2}\cdots b_1b_0)_B = \sum_{i=0}^{n-1} b_i \cdot B^i \qquad (1)$$

Regola 1:

1. si esprimono le cifre b_i e la base *B* nella base *B'* (solitamente banale)
2. si calcola la sommatoria *(1)*

Esempi

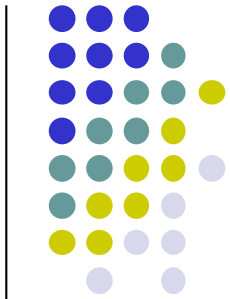


1. $10011_2 = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 =$
   $= 1 + 2 + 16 = 19$

2. $237_8 = 7 \cdot 8^0 + 3 \cdot 8^1 + 2 \cdot 8^2 = 7 + 24 + 128 = 159$

3. $1AF_{16} = 15 \cdot 16^0 + 10 \cdot 16^1 + 1 \cdot 16^2 =$
   $= 15 + 160 + 256 = 431$

Regola 2
(delle divisioni successive)

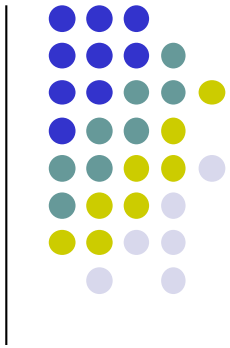


- Svolge le operazioni nella base di partenza *B,* per cui è molto adatta al caso in cui *B=10*
  - solitamente si usa per convertire da base 10 a qualsiasi altra base

- Si basa sull'osservazione che, dividendo il numero per *B'*:
  - il resto della divisione corrisponde alla cifra meno significativa del numero nella base *B'*
  - il quoziente al numero ottenuto cancellando la cifra meno significativa dal numero di partenza espresso in base *B'*
  - le cifre più significative possono essere quindi determinate riapplicando ricorsivamente lo stesso metodo al quoziente

Esempio

357:10 è pari a *35* con resto *7*
35:10 è pari a *3* con resto *5*
3:10 è pari a *0* con resto *3*



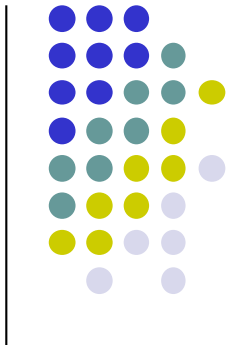
Regola 2:

1. si determinano il quoziente ed il resto della divisione del numero per *B'*
2. si prosegue come al passo *1*. considerando di volta in volta come numero di partenza il quoziente della divisione effettuata nel passo precedente, finché non si determina un quoziente nullo
3. si scrivono tutti i resti ottenuti in ordine inverso, esprimendoli nella base B' (solitamente banale)

NB:

- il primo resto ottenuto (al passo 1.) corrisponde alla cifra meno significativa, mentre l'ultimo a quella più significativa
- se *B'<B* tutti i resti ottenuti sono già espressi nella base *B'*
- altrimenti, sono numeri in base *B* che corrispondono direttamente a cifre in base *B'*

Esempi



1) *37 (B=10 e B'=2)*

37:2 = 18 con resto 1
18:2 = 9 con resto 0
9:2 = 4 con resto 1
4:2 = 2 con resto 0
2:2 = 1 con resto 0
*1:2 = **0** con resto 1*

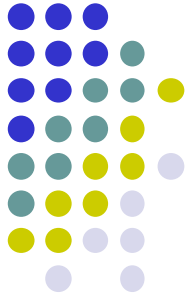
Quindi *37=100101_2*

2) *3226 (B=10 e B'=16)*

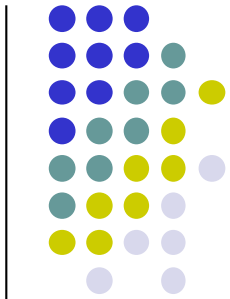
3226:16 = 201 con resto 10 (A)
201:16 = 12 con resto 9 (9)
*12:16 = **0** con resto 12 (C)*

Quindi *3226=$C9A_{16}$*

Osservazioni



- Per comprendere pienamente le precedenti regole ed esempi, osserviamo che:
  - occorre tenere sempre a mente la differenza tra quantità numeriche coinvolte e modo di rappresentarle
  - le quantità coinvolte infatti sono sempre le stesse
  - la loro rappresentazione (sintassi) può invece variare dipendentemente dal sistema di numerazione che consideriamo
- Rivediamo l'esempio precedente svolto in basi diverse:



1) *3226 (B=10 e B'=16)*

3226:16 = 201 con resto 10 (A)
201:16 = 12 con resto 9 (9)
*12:16 = **0** con resto 12 (C)*

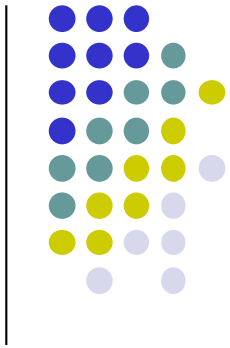
2) $C9A_{16}$ *(B=10 e B'=16)*

$C9A_{16}:10_{16} = C9_{16}$ *con resto* A_{16}
$\underline{C}9_{16}:10_{16} = C_{16}$ *con resto* 9_{16}
$\underline{C}_{16}:10_{16} = 0_{16}$ *con resto* C_{16}

Quindi *3226*=$C9A_{16}$

NB:

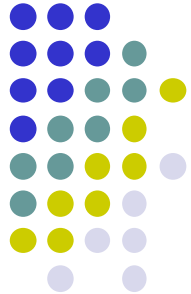
- i resti (e le altre quantità) sono identici, indipendentemente dal sistema utilizzato per effettuare i calcoli
- non conoscendo le rappresentazioni in base 16 del lato 2), e sapendo effettuare i calcoli in base 10, noi determiniamo i resti come in 1)



- In generale, per passare da base B a base $B' \neq B$ con $B \neq 10$ e $B' \neq 10$:
  1. si trasforma il numero da base B a base 10 usando la regola 1
  2. si trasforma il numero da base 10 a base B' usando la regola 2

- Se invece si vuole passare da base B a base B^k (cioè una potenza di B) o viceversa, esiste un metodo più semplice e diretto

Conversione da base B a B^k



- Partendo dalla cifra meno significativa si raggruppano le cifre in *k-uple*
- Si trasforma ogni *k*-upla nella relativa cifra in base B^k

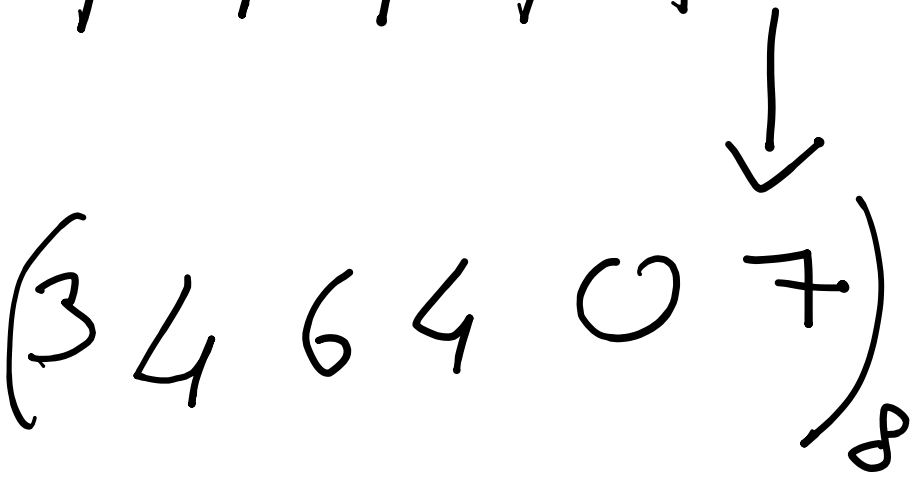
ESEMPIO: conversione da base 2 a base 2^4=16 (k=4)

1110011010000111_2  1110011010000111_2

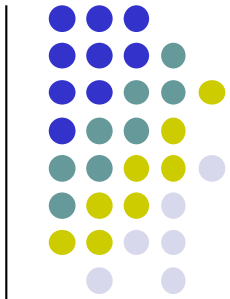
$0111 = 7_{16}$
$0000 = 0_{16}$
$1101 = D_{16}$
$1100 = C_{16}$
$1 = 1_{16}$

$$1110011010000111_2 = 1CD07_{16}$$

011/100/110/100/000/111 in base 8



Conversione da base B^k a B



- Partendo dalla cifra meno significativa si trasforma ogni cifra del numero in base B^k nelle corrispondenti k cifre della base B

ESEMPIO: conversione da base $8=2^3$ a base 2 ($k=3$)

56731_8

$1 = 001_2$
$3 = 011_2$
$7 = 111_2$
$6 = 110_2$
$5 = 101_2$

$$56731_8 = 101110111011001_2$$

k bit

2^k numeri naturali

$[0, 2^k - 1]$