

ARCHITETTURA DEGLI ELABORATORI

A.A. 2020-2021

Università di Napoli Federico II
Corso di Laurea in Informatica

Docenti

Proff.

Luigi Sauro gruppo 1 (A-G)

Silvia Rossi gruppo 2 (H-Z)



Rappresentazione dell'informazione

- Informazione: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere
 - Consente di ridurre l'incertezza
- Dato: elementi di informazione costituiti da simboli che debbono essere elaborati

Consideriamo un mazzo di carte

- E peschiamo una carta a caso senza guardarla
 - La carta potrebbe essere una qualsiasi delle 56 a disposizione
- Se vediamo che è una carta di cuori
 - Le possibilità si riducono a 13
- La carta è un asso di cuori

Rappresentazione dell'informazione

- In generale ogni rappresentazione è una funzione che associa ad ogni elemento una sequenza di simboli
- Per ogni rappresentazione, oggetti (numeri) distinti devono avere differenti rappresentazioni e la rappresentazione di ogni oggetto deve essere **unica e non ambigua**
 - pesca
- Le rappresentazioni usate sui calcolatori impiegano tutte sequenze finite di simboli, tali quindi da rappresentare insiemi finiti di naturali

The Digital Abstraction

Most physical variables are
continuous

Voltage on
a wire

Frequency
of an
oscillation

Position of
a mass



Digital abstraction considers
discrete subset of values

Alfabeti, parole, linguaggi

- Per alfabeto A intenderemo un insieme finito e distinguibile di segni che chiameremo a seconda del contesto cifre, lettere, caratteri, simboli etc.
 - Le nove cifre dell'alfabeto decimale $A=\{'0',\dots,'9'\}$
 - Le ventisei lettere (minuscole) dell'alfabeto $A=\{'a',\dots,'z'\}$
 - I quattro simboli delle carte francesi $A=\{\clubsuit,\diamond,\heartsuit,\spadesuit\}$
- Una parola (o stringa) su A è una sequenza finita di simboli dell'alfabeto A
 - “18945” ($A=\{'0',\dots,'9'\}$)
 - “pkwocod” ($A=\{'a',\dots,'z'\}$)
 - “ $\clubsuit\ \clubsuit\ \diamond\ \spadesuit$ ” ($A=\{\clubsuit,\diamond,\heartsuit,\spadesuit\}$)
- Con A^* indicheremo tutte le possibili parole generabili a partire dall'alfabeto A

Alfabeti, parole, linguaggi

Un linguaggio L sull'alfabeto A è un qualsiasi sottoinsieme di A^* .

Parole italiane

- “pwfnfkr”
- “dog”
- “siengs”
- “**casa**”
- “door”
- “**porta**”

Parole inglesi

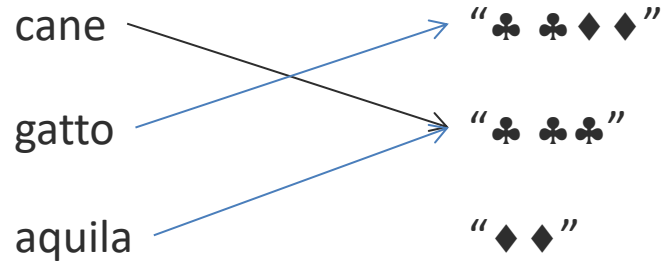
- “pwfnfkr”
- “**dog**”
- “siengs”
- “casa”
- “**door**”
- “porta”

Codifica, decodifica

- Le parole di un linguaggio non sono altro che sequenze di simboli che non hanno alcun senso finché non forniamo un modo per interpretarle
 - “♣ ♣ ♦ ♠” ?
- Associare gli elementi di un insieme D alle parole di un linguaggio L viene detta codifica o rappresentazione di D.
- Ad esempio sia D l'insieme dei concetti cane, gatto e aquila (notate bene che sto parlando dei concetti non delle parole “cane”, “gatto” e “aquila”). Allora nella usuale codifica della lingua inglese
 - Cane → “dog”
 - Gatto → “cat”
 - Aquila → “eagle”
- Formalmente una codifica è una funzione totale $f:D \rightarrow L$
 - Se f non è suriettiva allora diremo che è ridondante (1 a 1)
 - Se f non è iniettiva allora diremo che è ambigua (più di 1 a 1)

Codifica, decodifica

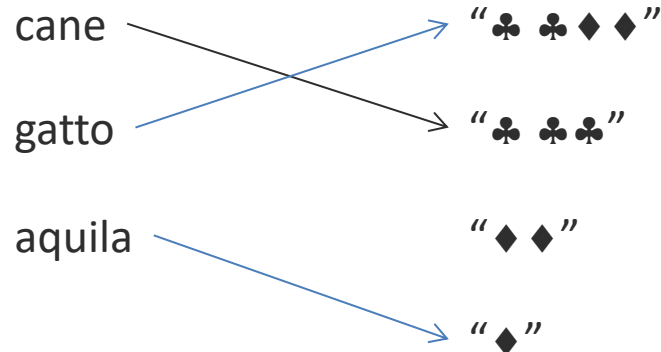
- Sia $D=\{\text{cane, gatto, aquila}\}$ e $L=\{\text{"♣ ♣ ♦ ♦"}, \text{"♣ ♣ ♣"}, \text{"♦ ♦"}\}$ e sia f rappresentata graficamente



- f è ambigua e ridondante

Codifica, decodifica

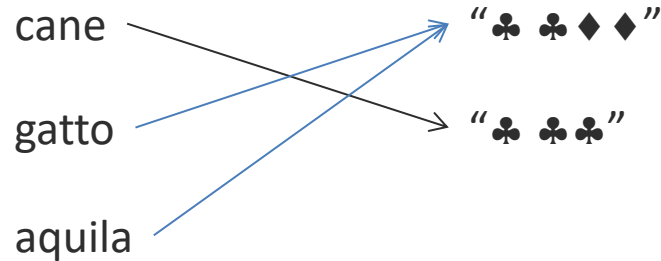
- Sia $D=\{\text{cane, gatto, aquila}\}$ e $L=\{“\clubsuit \clubsuit \spadesuit \spadesuit”, “\clubsuit \clubsuit \clubsuit”, “\spadesuit \spadesuit”, “\spadesuit”\}$ e sia f rappresentata graficamente



- f non è ambigua ma è ridondante

Codifica, decodifica

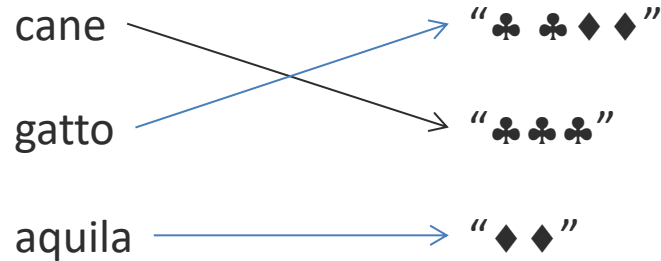
- Sia $D=\{\text{cane, gatto, aquila}\}$ e $L=\{“\clubsuit \clubsuit \spadesuit \spadesuit”, “\clubsuit \clubsuit \clubsuit”\}$ e sia f rappresentata graficamente



- f non è ridondante ma è ambigua

Codifica, decodifica

- Sia $D=\{\text{cane, gatto, aquila}\}$ e $L=\{\text{"♣ ♣ ♦ ♦"}, \text{"♣ ♣ ♣"}, \text{"♦ ♦"}\}$ e sia f rappresentata graficamente



- f non è ambigua e non è ridondante

Codifica, decodifica

- Sia N_D la cardinalità di D e N_L la cardinalità di L
 - Se $N_D > N_L$ qualsiasi codifica $f: D \rightarrow L$ è ambigua
 - Se $N_D < N_L$ qualsiasi codifica $f: D \rightarrow L$ è ridondante
- Inoltre è bene notare che
 - Non ambiguo significa che f è iniettiva (ad elementi distinti di D corrispondono elementi distinti di L)
 - Non ridondante significa che f è suriettiva (ogni elemento di L è la codifica di almeno un elemento di D)
 - Quindi se f è non ambigua e non ridondante allora f è iniettiva e suriettiva, quindi è una funzione biunivoca
- Una funzione $g: L \rightarrow D$ è invece detta una decodifica di D
- Se f è non ambigua (ovvero iniettiva) allora è invertibile. Quindi, induce naturalmente una decodifica $g=f^{-1}$

Proprietà delle codifiche

- Abbiamo già visto
 - ambigua/non ambigua
 - ridondante/non ridondante
- Altre proprietà
 - Economicità: numero di simboli utilizzati per unità di informazione
 - Semplicità nell'operazione di codifica e decodifica
 - Semplicità nell'eseguire operazioni sull'informazione codificata

Rappresentazione dei numeri naturali

- Occupiamoci ora delle possibili codifiche dei numeri naturali $0, 1, 2, \dots$
- Un codice ovvio è dato dal sistema di numerazione decimale basato su $A = \{ '0', \dots, '9' \}$.
- Tale sistema è un sistema posizionale ovvero ad ogni cifra di una parola è assegnato un peso differente a seconda della posizione nella sequenza. Ad esempio dato "4456"
 - '6' rappresenta sei unità (cifra meno significativa)
 - '5' rappresenta cinque decine
 - '4' rappresenta quattro centinaia
 - '4' rappresenta quattro migliaia (cifra più significativa)

Fin'ora ho estensivamente usato gli apici per distinguere le parole ("4456") da quello che rappresentano (il numero 4456). E' chiaro che usare gli apici ogni volta per rimarcare la distinzione fra numerali (parole) e numeri (concetti) può rendere molto pesante la trattazione. Quindi in seguito inizierò ad omettere gli apici qualora il contesto non generi ambiguità.

Il sistema numerico decimale è un sistema di tipo **posizionale** ovvero:

Le cifre che compongono un numero cambiano il loro valore secondo la posizione che occupano

7237 (settemiladuecentotrentasette) in base 10

$$\begin{array}{rcll} 7 \times 10^3 + & 2 \times 10^2 + & 3 \times 10^1 + 7 \times 10^0 \\ 7 \times 1000 + & 2 \times 100 + & 3 \times 10 + 7 \times 1 \\ 7000 + & 200 + & 30 + 7 & = \mathbf{7237} \end{array}$$

colonna dell'1
colonna del 10
colonna del 100
colonna del 1000

$$9742_{10} = 9 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$$

nove sette quattro due
1000 100 10 1

Rappresentazione in base 10

- Decomponendo in potenze di 10, il numerale 1024 rappresenta il numero

$$1 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$$

- Generalizzando, un numerale $c_{m-1}c_{m-2} \dots c_0$ rappresenta

$$\sum_{i=0}^{m-1} c_i \cdot 10^i$$

- Il sistema decimale è quindi una codifica posizionale su base 10. Tuttavia non è l'unica, ad esempio i babilonesi utilizzavano un sistema di numerazione su base 60 (sessagesimale)

Number Systems

- Decimal numbers

1's column
10's column
100's column
1000's column

$$5374_{10} =$$

- Binary numbers

1's column
2's column
4's column
8's column

$$1101_2 =$$



Number Systems

- Decimal numbers

1's column
10's column
100's column
1000's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five three seven four
thousands hundreds tens ones

- Binary numbers

1's column
2's column
4's column
8's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one one no one
eight four two one



Powers of Two

- $2^0 =$

- $2^1 =$

- $2^2 =$

- $2^3 =$

- $2^4 =$

- $2^5 =$

- $2^6 =$

- $2^7 =$

- $2^8 =$

- $2^9 =$

- $2^{10} =$

- $2^{11} =$

- $2^{12} =$

- $2^{13} =$

- $2^{14} =$

- $2^{15} =$



Powers of Two

- $2^0 = 1$

- $2^1 = 2$

- $2^2 = 4$

- $2^3 = 8$

- $2^4 = 16$

- $2^5 = 32$

- $2^6 = 64$

- $2^7 = 128$

- $2^8 = 256$

- $2^9 = 512$

- $2^{10} = 1024$

- $2^{11} = 2048$

- $2^{12} = 4096$

- $2^{13} = 8192$

- $2^{14} = 16384$

- $2^{15} = 32768$

**Handy to
memorize
up to 2^9**



Rappresentazione

- Il sistema **decimale** utilizza **dieci** simboli per rappresentare un numero

- 0 1 2 3 4
 5 6 7 8
 9

- Il sistema **binario** utilizza **due** simboli

- 0 1

- Il sistema **ottale** utilizza **otto** simboli

- 0 1 2 3 4
 5 6 7

- Il sistema **esadecimale** utilizza **sedici** simboli

- 0 1 2 3 4 5 6 7 8 9
 A B C D E F

Rappresentazione in una base generica

- Ad ogni naturale $b > 1$ corrisponde una codifica in base b .
- L'alfabeto A_b consiste in b simboli distinti che corrispondono ai numeri $0, 1, \dots, b-1$.
- Analogamente al sistema decimale, un numerale di cifre di A_b rappresenta il numero $s_{m-1} \dots s_0$

$$\sum_{i=0}^{m-1} s_i \cdot b^i$$

$$587_{10} = 5 \times 10^2 + 8 \times 10^1 + 7 \times 10^0$$

Consideriamo ad esempio il sistema ottale (base 8). A_8 consiste nelle cifre '0','1',...,'7'

$$13_{\underset{8}{}} \longrightarrow 1 \cdot 8^1 + 3 \cdot 8^0 = 8 + 3 = 11$$

$$5201_{\underset{8}{}} \longrightarrow 5 \cdot 8^3 + 2 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 = 5 \cdot 8^3 + 2 \cdot 8^2 + 1 \cdot 8^0 = 2560 + 128 + 1 = 2689$$

Rappresentazione in una base generica

A questo punto potrebbero sorgere delle ambiguità. Se qualcuno vi dicesse: 11 è un numero pari. Pensereste che sia impazzito. Lui potrebbe ribattere: certo! infatti in base 5

$$11 \longrightarrow 1 \cdot 5^1 + 1 \cdot 5^0 = 5 + 1 = 6$$

E' chiaro che bisogna mettersi d'accordo su quale sistema di numerazione si adotta di volta in volta. Per questo useremo delle notazioni standard

- n denota un (generico) numero naturale, a prescindere dalla sua rappresentazione
- n_b denota un naturale rappresentato in base b
- Una sequenza di cifre decimali rappresenta un particolare naturale espresso in base dieci. Ad esempio 236 denota il numero duecentotrentasei
- Una sequenza di cifre seguite da un pedice b rappresenta un numero espresso in base b .
- esempio:

$$1001_2 = 2^3 + 1 = 9$$

Basi maggiori di 10

- Per basi $b < 10$ possiamo chiaramente ri-usare le usuali cifre '0',..., '9'. Ad esempio, la codifica in base 6 utilizza le cifre '0',..., '5' mentre quella in base 3 le cifre '0', '1' e '2' e così via.
- E per basi $b > 10$?
 - Si prendono in prestito le lettere dell'alfabeto
 - Per $b=16$ le cifre adottate sono '0',..., '9', 'a',..., 'f', dove:

$$a_{16} = 10 \quad b_{16} = 11$$

$$c_{16} = 12 \quad d_{16} = 13$$

$$e_{16} = 14 \quad f_{16} = 15$$

Esempio:

$$b3c_{16} = 11 \cdot 16^2 + 3 \cdot 16^1 + 12 \cdot 16^0 = 2816 + 48 + 12 = 2876$$

colonna dell'1
colonna del 16
colonna del 256

$$2ED_{16} = \underset{\substack{\text{due} \\ 256}}{2} \times 16^2 + \underset{\substack{\text{quattordici} \\ 16}}{E} \times 16^1 + \underset{\substack{\text{tredici} \\ 1}}{D} \times 16^0 = 749_{10}$$

101100

bit più
signifi-
cativo

bit meno
signifi-
cativo

(a)

DEAFDAD 8




























































byte più
signifi-
cativo

byte meno
signifi-
cativo

(b)

Basi maggiori di 10

E se le lettere dell'alfabeto non dovessero bastare?

 1	 11	 21	 31	 41	 51
 2	 12	 22	 32	 42	 52
 3	 13	 23	 33	 43	 53
 4	 14	 24	 34	 44	 54
 5	 15	 25	 35	 45	 55
 6	 16	 26	 36	 46	 56
 7	 17	 27	 37	 47	 57
 8	 18	 28	 38	 48	 58
 9	 19	 29	 39	 49	 59
 10	 20	 30	 40	 50	

Lunghezza di n rispetto ad una base

- Chiamiamo lunghezza di n rispetto a b il numero di cifre che occorrono per rappresentare n in base b
 - la lunghezza di 101 rispetto a 2 è 7:
 $1100101_2 = 101$
 - la lunghezza di 101 rispetto a 10 è 3:
 $101_{10} = 101$
 - la lunghezza di 101 rispetto a 16 è 2:
 $65_{16} = 101$
- La lunghezza di un numerale decresce al crescere della base di codifica

Valore massimo rappresentabile

Riferendoci alla base 10

- Con 1 cifra rappresentiamo i numeri da 0 a 9
- Con 2 cifre i numeri da 0 a 99
- Con 3 cifre i numeri da 0 a 999
- Con m cifre i numero da 0 a 10^m-1

Quindi se indichiamo con v_{\max} il maggior numero rappresentabile con m cifre in base 10 abbiamo

$$v_{\max} = 10^m - 1$$

Valore massimo rappresentabile

- E per una base diversa da 10 quale è il massimo valore rappresentabile con m cifre?
- Consideriamo, ad esempio, il caso $b=2$ e $m=4$, il massimo valore rappresentabile corrisponde al numerale 1111

$$1111_2 = 2^3 + 2^2 + 2^1 + 2^0 = 15 = 2^4 - 1$$

- Per una generico b e m il maggior numero rappresentabile si ottiene concatenando m volte la cifra “più alta” ($b-1$). Quindi:

$$\begin{aligned} v_{max} &= (b-1)b^{m-1} + (b-1)b^{m-2} + \dots + (b-1)b^1 + (b-1)b^0 = \\ &= (b \cdot b^{m-1} - b^{m-1}) + (b \cdot b^{m-2} - b^{m-2}) + \dots + (b \cdot b^1 - b^1) + (b \cdot b^0 - b^0) = \\ &= b^m - b^{m-1} + b^{m-1} - b^{m-2} + \dots + b^2 - b + b - 1 = b^m - 1 \end{aligned}$$

$$b^m - 1 \geq n$$

$$b^m \geq n + 1$$

$$m \geq \log_b(n + 1)$$

$$m \geq \log_b(n + 1)$$

$$m = \lceil \log_b(n + 1) \rceil$$

Cifre necessarie per rappresentare n

- Nella slide precedente avevamo il numero di cifre m e volevamo sapere quale è il *massimo numero rappresentabile* in una certa base b .
- Consideriamo il problema inverso: abbiamo un valore n e ci chiediamo quante *cifre m occorrono per rappresentarlo*.
- Chiaramente il massimo numero rappresentabile con m cifre dovrà essere maggiore o uguale a n
- In particolare cerchiamo il più piccolo m tale che

Binary Values and Range

- **N -digit decimal number**
 - How many values? 10^N
 - Range? $[0, 10^N - 1]$
 - Example: 3-digit decimal number:
 - $10^3 = 1000$ possible values
 - Range: $[0, 999]$
- **N -bit binary number**
 - How many values? 2^N
 - Range: $[0, 2^N - 1]$
 - Example: 3-digit binary number:
 - $2^3 = 8$ possible values
 - Range: $[0, 7] = [000_2 \text{ to } 111_2]$



Digressione: base unaria

- Ricordate che quando abbiamo introdotto i sistemi di numerazione abbiamo assunto la base $b \geq 2$.
- Perché non è possibile considerare una numerazione unaria? Certo! Ma ci sono delle controindicazioni...
 - La base unaria consta di una solo cifra I (detta in gergo matematico *mazzarella* 😊)
 - Una mazzarella rappresenta 0, due mazzarelle 1, ..., n+1 mazzarelle rappresentano n
 - $IIIII_1 = 5$
 - Notate che a prescindere dalla posizione ogni mazzarella vale 1, quindi questo sistema non è posizionale
 - Non potrebbe essere altrimenti visto che 1 elevato ad una qualsiasi potenza è sempre uguale a 1!

Codifica ottimale

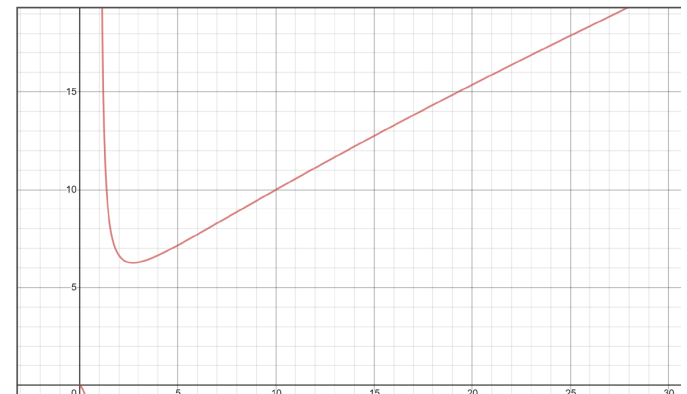
- Ritorniamo alle basi ammissibili ($b \geq 2$)
- Abbiamo visto che più grande è la base b , minore è il numero di cifre che occorrono per rappresentare un numero
- Quindi il codice binario è il sistema di codifica meno economico fra quelli ammissibili
 - Per esempio $\log_2 10 \cong 3.32$ questo vuol dire che per rappresentare un numero n in binario occorrono *circa il triplo* delle cifre che occorrono per rappresentare lo stesso n nel sistema decimale
 - *Perché i computer adottano la codifica binaria?*

Codifica ottimale

- Non bisogna tener presente solo la lunghezza di codifica ma anche il fatto che un calcolatore per operare in una certa base b deve poter rappresentare tutte le cifre di quella base, *quindi gli servono b differenti stati*.
- Una nozione di costo di una codifica che tiene conto anche di questo fattore è data dal prodotto

$$b \cdot m_b \simeq b \cdot \log_b n$$

- Si può mostrare che il valore di b che minimizza tale costo è il numero di Nepero $e \cong 2.7$, quindi le codifiche che si avvicinano di più a tale valore ottimale sono quelle in base 2 e 3



Digital Discipline : Binary Values



Two discrete values:

1's and 0's
1, TRUE, HIGH
0, FALSE, LOW



1 and 0: voltage levels, rotating gears, fluid levels, etc.



Digital circuits use **voltage** levels to represent 1 and 0



Bit: Binary digit

Esempio: codificare 251_{10}
in base 3

$$251/3 = 83 \quad \text{resto: } 2$$

$$83/3 = 27 \quad \text{resto: } 2$$

$$27/3 = 9 \quad \text{resto: } 0$$

$$9/3 = 3 \quad \text{resto: } 0$$

$$3/3 = 1 \quad \text{resto: } 0$$

$$1/3 = 0 \quad \text{resto: } 1$$

$$251_{10} = 100022_3$$

Cambiamento di base

- Problema: dato n rappresentato in base a , quale è la sua rappresentazione in base b ?
- Supponiamo di saper fare le quattro operazioni in base a
- L'algoritmo di cambiamento di base consiste nel dividere ripetutamente n (espresso in base a) per b finché il quoziente non risulti uguale a zero. La sequenza di resti ottenuti (compresi tra 0 e $b-1$) è la codifica dalla cifra meno significativa a quella più significativa di n in base b

$43 : 2 = 21$	con resto di 1
$21 : 2 = 10$	con resto di 1
$10 : 2 = 5$	con resto di 0
$5 : 2 = 2$	con resto di 1
$2 : 2 = 1$	con resto di 0
$1 : 2 = 0$	con resto di 1

$$43 = 101011$$

Cambiamento di base

- Esempio: codificare 333_7 in base 9
- Problema: non siamo molto allenati con la divisione in base 7. Meglio affrontare il problema in due passi:

– Codifico 333_7 in base 10

$$333_7 = 3 \cdot 7^2 + 3 \cdot 7^1 + 3 \cdot 7^0 = 171$$

– Codifico 171_{10} in base 9

$$171/9 = 19 \quad \text{resto: } 0$$

$$19/9 = 2 \quad \text{resto: } 1$$

$$2/9 = 0 \quad \text{resto: } 2$$

$$333_7 = 210_9$$

Decimal to Binary Conversion

- Two methods:
 - **Method 1:** Find the largest power of 2 that fits, subtract and repeat
 - **Method 2:** Repeatedly divide by 2, remainder goes in next most significant bit



Decimal to Binary Conversion

Method 1: Find the largest power of 2 that fits, subtract and repeat

53_{10}

Method 2: Repeatedly divide by 2, remainder goes in next most significant bit

53_{10}



Decimal to Binary Conversion

Method 1: Find the largest power of 2 that fits, subtract and repeat

53_{10}	32×1
$53 - 32 = 21$	16×1
$21 - 16 = 5$	4×1
$5 - 4 = 1$	1×1

$$= 110101_2$$

Method 2: Repeatedly divide by 2, remainder goes in next most significant bit

$53_{10} =$	$53/2 = 26$	R1
	$26/2 = 13$	R0
	$13/2 = 6$	R1
	$6/2 = 3$	R0
	$3/2 = 1$	R1
	$1/2 = 0$	R1

$$= 110101_2$$



Number Conversion

- Binary to decimal conversion:
 - Convert 10011_2 to decimal
- Decimal to binary conversion:
 - Convert 47_{10} to binary



Number Conversion

- Binary to decimal conversion:
 - Convert 10011_2 to decimal
 - $16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 19_{10}$
- Decimal to binary conversion:
 - Convert 47_{10} to binary
 - $32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 = 101111_2$



Decimal to Binary Conversion

Another example: Convert 75_{10} to binary.



Decimal to Binary Conversion

Another example: Convert 75_{10} to binary.

$$75_{10} = 64 + 8 + 2 + 1 = 1001011_2$$



Decimal to Binary Conversion

Another example: Convert 75_{10} to binary.

$$75_{10} = 64 + 8 + 2 + 1 = 1001011_2$$

or

$$75/2 = 37 \quad R1$$

$$37/2 = 18 \quad R1$$

$$18/2 = 9 \quad R0$$

$$9/2 = 4 \quad R1$$

$$4/2 = 2 \quad R0$$

$$2/2 = 1 \quad R0$$

$$1/2 = 0 \quad R1$$



Codifica binaria e esadecimale

- Abbiamo detto che i componenti digitali operano in codice binario.
- Tuttavia la rappresentazione in binario non è molto human-friendly perché genera codici piuttosto lunghi
 - $599 = 1001010111_2$
- Si potrebbe pensare di usare la nostra base naturale (10). Questo comporta usare il precedente algoritmo per codifica/decodifica
 - Non proprio agevole
 - Non proprio velocissimo
 - Il problema è che 10 non è una potenza di 2

Codifica binaria e esadecimale

- La codifica/decodifica in una base m potenza di 2 permette invece di usare qualche trucchetto che migliora i tempi di codifica e decodifica.
- le cifre utilizzate nel sistema esadecimale sono '0',..., '9', 'a',..., 'f'.
- Inoltre, poiché $16=2^4$ è possibile codificare ogni cifra del sistema esadecimale mediante 4 bit
- Viceversa 4 bit del sistema binario corrispondono ad una cifra esadecimale

Codifica binaria e esadecimale

- Codifica delle cifre esadecimali in binario

0 \longrightarrow 0000

1 \longrightarrow 0001

2 \longrightarrow 0010

3 \longrightarrow 0011

4 \longrightarrow 0100

5 \longrightarrow 0101

6 \longrightarrow 0110

7 \longrightarrow 0111

8 \longrightarrow 1000

9 \longrightarrow 1001

a \longrightarrow 1010

b \longrightarrow 1011

c \longrightarrow 1100

d \longrightarrow 1101

e \longrightarrow 1110

f \longrightarrow 1111

- Notate che questa codifica corrisponde alla codifica dei rispettivi numeri con possibili 0 non significativi

$$0111_2 = 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 7$$

Da esadecimale a binario

- Dato un numerale esadecimale per codificarlo in binario è sufficiente giustapporre le codifiche delle singole cifre (eliminando eventualmente zeri non significativi)
- Esempi:
 - $4c9f_{16} \rightarrow 0100\ 1100\ 1001\ 1111 \rightarrow 10011001001111_2$
 - $b2a_{16} \rightarrow 1011\ 0010\ 1010 \rightarrow 101100101010_2$
 - $157_{16} \rightarrow 0001\ 0101\ 0111 \rightarrow 101010111_2$
- Nota bene che $157_{16} = 1 \cdot 16^2 + 5 \cdot 16^1 + 7 \cdot 16^0 = 343$

Da binario a esadecimale

- Per il passaggio di base da binario a esadecimale si effettua la codifica inversa avendo cura di aggiungere eventuali zeri non significativi in modo che la lunghezza del numerale in binario sia multipla di 4
- Esempi:
 - $10_2 \rightarrow 0010 \rightarrow 2_{16}$
 - $10011_2 \rightarrow 0001\ 0011 \rightarrow 13_{16}$
 - $1111100101011100_2 \rightarrow 1111\ 1001\ 0101\ 1100 \rightarrow f95c_{16}$

Nota: comunemente un numerale esadecimale viene indicato con il prefisso 0x

$f95c_{16} \rightarrow 0xf95c$

Rappresentazione registri

- Un computer le grandezze numeriche sono elaborate mediante sequenze di simboli di lunghezza fissa dette parole
- Poichè una cifra esadecimale codifica 4 bit:
 - 1 byte (8 bit) → 2 cifre esadecimali
 - 4 byte (32 bit) → 8 cifre esadecimali
 - 8 byte (64 bit) → 16 cifre esadecimali

Bits, Bytes, Nibbles...

- Bits

10010110
most significant bit least significant bit

- Bytes & Nibbles

byte
10010110
nibble

- Bytes

CEBF9AD7
most significant byte least significant byte



Word

- I microprocessori gestiscono gruppi di bit chiamati word
 - La grandezza dipende dall'architettura del microprocessore
 - 64 bit (o 32)
 - 10011 -> 0001 0011