Sistemes Informàtics

UF1. Instal·lació, configuració i explotació del sistema informàtic.

NF1.1

Anàlisi i instal·lació de sistemes informàtics

Sistemes numèrics i portes lògiques

Introducció

Un **sistema digital** és qualsevol sistema de transmissió o processament d'informació en el qual la informació es representa mitjançant **valors discrets**.

Un **bit** (***bi**nary digit*) diferència **dos estats**, és la quantitat mínima d'informació, i sovint s'associa amb els conceptes de cert/fals, o en electrònica com a obert/tancat. Com que és fàcil realitzar components físics capaços de diferenciar dos estats, la **base 2** és la més utilitzada per a les variables discretes. El sistema matemàtic que utilitza dos dígits és anomenat **sistema binari**.

Els fonaments dels sistemes binaris van ser establerts pel matemàtic britànic **George Boole**.

Tipus de sistemes digitals

Sistemes combinacionals: la sortida només depèn de l'entrada actual

$$y(t_i) = F\left[x(t_i)\right]$$

Sistemes seqüencials: la sortida només depèn de l'entrada actual i de la història passada

$$y(t_i) = F\left[x(t_i), x(-\infty, t_i)\right]$$

Sistema de numeració

Un sistema de numeració és un conjunt de símbols i regles de generació. Ho podem representar com N = S + R on

- **N** es **sistema de numeració** considerat
- **S** són els **símbols** permesos al sistema.
  - En decimal {0,1,2,3,4,5,6,7,8,9}
  - En binari {0,1}
  - En octal {0,1,2,3,4,5,6,7}
  - En hexadecimal {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
- **R** son les **regles de generació** que ens indiquen quins nombres són vàlids i quins no.
  - Les regles per als sistemes posicionals són les mateixes en tots els sistemes, només canvia la base. Els sistemes no posicionals, com el romà, tenen regles més complexes.

Sistemes de numeració posicionals

Base b

$$N = \underbrace{a_n b^n + a_{n-1} b^{n-1} + \ldots + a_0 b^0}_{\text{part entera}} + \underbrace{a_{-1} b^{-1} + \ldots + a_{-p} b^{-p}}_{\text{part fraccionària}}$$

$0 \leq a_i < b$

n + 1 és el nombre de dígits enters
p és el nombre de dígits fraccionaris

$$1385 = \mathbf{1} \cdot 10^3 + \mathbf{3} \cdot 10^2 + \mathbf{8} \cdot 10^1 + \mathbf{5} \cdot 10^0$$

$$1385 = \mathbf{1} \cdot 1000 + \mathbf{3} \cdot 100 + \mathbf{8} \cdot 10 + \mathbf{5} \cdot 1$$

Sistemes de numeració posicionals comuns

El sistema binari és de gran importància per ser fàcilment implementable en el maquinari com a dos estats (on/off)

Els sistemes més usats, a més del decimal (b=10) i el binari (b=2), són l'octal (b=8) i l'hexadecimal (b=16).

Hexadecimal i octal són molt útils perquè es poden representar en base 2 ($2 = 2^1$, $8 = 2^3$, $16 = 2^4$)

Això permet agrupar els nombres en **grups de tres** (***octal***) i **quatre bits** (***hexadecimal***) i convertir-los directament.

Un **byte** conté 8 bits, que es poden representar en **dos dígits hexadecimals**, per això és molt més freqüent que l'octal.

Decimal	Binari	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

El bit i el byte

Cada **digit del sistema binari** pot tenir els valors 0 o 1. Els digits binaris reben el nom de **bit**, abreviació de ***bi**nary-digi**t***. És la **unitat mínima d'informació**.

Un **byte** són 8 bits, i en les computadores actuals és la **unitat mínima d'emmagatzematge**, i la unitat mínima que pot adreçar-se. El seu origen està en el codi ASCII i l'ús de computadores per àmbits generals; les computadores de la URSS (*Unió Soviètica*) tenien 40, 42, 43 bits...

Ha quedat demostrada la conveniencia del byte, per ser una potència de 2.

$$2^3 = 8$$

Sovint es fa servir ***b*** com a símbol del bit, tot i no ser una abreviació reconeguda: ***bit*** ja és una abreviació, i no hauria d'abreviar-se més.

També de manera informal es fa servir ***B*** com a símbol del ***byte***, però tot i que tampoc està reconeguda, en general sí que es diferència ***b*** per a bit (generalment en comunicacions) i ***B*** per a byte.

És important ser conscient de quan ens referim a un o a l'altre.

Unidad de medida	Símbolo	Relación
bit	bit	1 bit
Byte	B	8 bits

Codificació

Les computadores tenen una longitud finita per als números; si un número excedeix aquesta longitud es produeix un **overflow**.

Les computadores fan servir el byte (8 bits) com a unitat mínima d'emmagatzematge, i la unitat mínima que pot adreçar-se.

En els anys 80 eren freqüents les computadores domèstiques de **8 bits**. Amb l'arribada del PC van passar a tenir **16 bits** i aviat van aparèixer els processadors Motorola 68000 i els Intel 80386 i 80486 (en 1989), ja de **32 bits**. El Pentium en 1993 va incorporar el bus de **64 bits**, i aviat els nous processadors treballaven amb operacions 64 bits.

També quan s'emmagatzemen les dades numériques es fa generalment en longitud fixa (*els formats de longitud variable són poc freqüents*).

Si tenim un número de longitud **L** podem representar b^L números, des del zero fins al $b^L - 1$.

En decimal, amb ***n*** digits podem codificar 10^n números diferents (recordar de comptar el zero).

En binari, fent servir ***n*** bits podem codificar fins al nombre 2^n-1, en total 2^n números.

Conversió de nombres

- Passar de **binari a decimal** (tutorial)
- Passar de **decimal a binari** (tutorial)
- Passar d'**hexadecimal a decimal** (tutorial)
- Passar de **decimal a hexadecimal** (tutorial)
- Passar d'**hexadecimal a binari**: immediat (*veure taula anterior*)
- Passar de **binari a hexadecimal**: immediat (*veure taula anterior*)

Exemples de conversió

Exemple convertir 524_{10} a base 2

524:2 = 262	**0**	=	a_0
262:2 = 131	**0**	=	a_1
131:2 = 65	**1**	=	a_2
65:2 = 32	**1**	=	a_3
32:2 = 16	**0**	=	a_4
16:2 = 8	**0**	=	a_5
8:2 = 4	**0**	=	a_6
4:2 = 2	**0**	=	a_7
2:2 = **1** = a_9	**0**	=	a_8

$524_{10} = 1000001100_2$

Exemple: convertir 1000001100_2 a base 10

1000001100_2

$= 0 \cdot 2^0 + 0 \cdot 2^1 + \mathbf{1} \cdot 2^2 + \mathbf{1} \cdot 2^3 + 0 \cdot 2^4 + 0 \cdot 2^5 + 0 \cdot 2^6 + 0 \cdot 2^7 + 0 \cdot 2^8 + \mathbf{1} \cdot 2^9$

$= 2^2 + 2^3 + 2^9$

$= 524_{10}$

Codificació binària

8 bits: 2^8 = 256. Entre zero i 255.

16 bits: 2^{16} = 65_536. Entre zero i 65_535.

32 bits: 2^{32} = 4_294_967_296.

64 bits: 2^{64} = 18_446_744_073_709_551_616.

Això en el processador representa la **magnitud dels números amb què pot operar en una única operació**, i en la seva connexió amb el bus de dades, la **màxima quantitat de memòria que pot direccionar**.

Representant **números negatius** (en complement a 2):

8 bits: Entre -2^7 i $2^7 - 1$ és a dir [-128 .. 127]

16 bits: Entre -2^{15} i $2^{15} - 1$, és a dir [-32768 .. 32767]

32 bits: Entre -2^{31} i $2^{31} - 1$.

64 bits: Entre -2^{63} i $2^{63} - 1$.

Complement a 2 (números negatius)

El complement a 2 del número *x* **fent servir N bits** es defineix com:

$C_2^x = 2^N - x$

Però el seu càlcul en la pràctica és més senzill:

L'obtenim sumant 1 de la representació en complement a 1 del número:

$C_2^x = C_1^x + 1$

On **el complement a 1** s'obté al canviar cada un dels seus elements pel seu complementari, és a dir, canviar els uns per zeros i els zeros per uns.

- El bit de l'esquerra serà sempre 1 per als valors negatius.
- Permet l'aprofitament dels 2^{N-1} valors.
  - És una representació *compacta*.
- El complement a 2 permet la representació amb un número finit de bits de números negatius, que són coherents per l'operació de suma amb la representació binària dels positius en el mateix número de bits.
  - És a dir, la resta es converteix en una suma del valor negatiu.
- Compleix la propietat simètrica:

$$C_2^{(C_2^x)} = x$$

Números binaris negatius: Complement a 2

Per la representació dels números negatius fem servir el complement a 2:

- El bit més significatiu indica el signe.
- El zero queda en el “conjunt de números positius”.
- La diferència entre dos números és la **suma binària del minuend amb el complement a dos del substraned**.
- És una operació simètrica.

Aquestes propietats fan que el complement a 2 sigui la representació més adequada per als números enters negatius.

Per **fer el complement a 2 d’un número**:

- Fem el complement a 1, canviant 1 per 0 i viceversa.
- Sumem 1.

Per exemple, **en 4 bits** el valor -5:

5 = 0101	C_1^5 = 1010	C_2^5 = 1011

El complement a 2 depèn del nombre de bits de la representació, en aquest cas 4. Fent servir més bits s’haurien d’extendre els bits significatius amb “1”.

Números decimals: punt fix

A més dels números enters és necessari codificar números decimals:

Punt fix: part entera i part decimal. Els números tenen representació finita en els computadors.

La capacitat de la part entera determina el **rang de números** representables, la capacitat de la part decimal indica la **precisió** amb que podem representar-los.

Quan la part entera és cero i la part decimal és tan petita que no pot representar-se es dona un ***underflow***, situació en la qual el resultat no és zero però ha quedat representat com a zero.

Compte! La representació en base 2 pot donar-nos resultats inesperats: `(1.1 + 2.2) == 3.3` **és fals**.

Això és degut al fet que els **nombres racionals** no poden representar-se com a nombres amb decimals finits; pe. en base 10 no podem representar exactament ⅓. De la mateixa manera, **en base 2 no es pot representar 1/10 de manera exacta**, és a dir, que el valor 0.1 en binari no es pot representar exactament.

Per evitar aquest problema molts càlculs financers fan servir **números decimals** (*BCD*) en comptes de binaris, tot i que és una característica avui dia en desús.

Unitats de mesura d'informació

Hi ha confusió respecte als símbols de les unitats de mesura de la informació, ja que no són part del SI.

La pràctica recomana que el **byte** es representi amb el símbol ***B* majúscula**, i el **bit**, tot i que no s'hauria d'abreujar, generalment amb el símbol ***b* minúscula**.

L'ús comercial ha provocat confusions: 1.024 (2^{10}) no és 1.000 (10^3)... les confusions començaren quan els discos ja tenien mides superiors a les 240Mb i la diferència començà a ser significativa.

10^3 = 1_000 : 2^{10} = 1_024 : 2^{20} = 1_048_576

240 * 2^{20} = 251_658_240 (dif: 11.658 MB)

Múltiples de bytes					
Prefix del SI (SI)			Prefix binari (IEC 60027-2)		
Nom	Símbol	Múltiple	Nom	Símbol	Múltiple
kilobyte	kB	10^3 (o 2^{10})	kibibyte	KiB	2^{10}
megabyte	MB	10^6 (o 2^{20})	mebibyte	MiB	2^{20}
gigabyte	GB	10^9 (o 2^{30})	gibibyte	GiB	2^{30}
terabyte	TB	10^{12} (o 2^{40})	tebibyte	TiB	2^{40}
petabyte	PB	10^{15} (o 2^{50})	pebibyte	PiB	2^{50}
exabyte	EB	10^{18} (o 2^{60})	exbibyte	EiB	2^{60}
zettabyte	ZB	10^{21} (o 2^{70})	zebibyte	ZiB	2^{70}
yottabyte	YB	10^{24} (o 2^{80})	yobibyte	YiB	2^{80}

Codificacions no estrictament posicionals

El codi binari natural és posicional: Amb *n* bits podem codificar fins al número 2^n-1

Decimal	Binari
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

El codi binari no té correspondència directa amb els dígits decimals; per això es fa servir l'octal i sobretot l'hexadecimal.

També existeix una codificació que estableix una correspondència directa amb els dígits binaris: *BCD* (*Binary Coded Decimal*). Presenta avantatges en algunes situacions, però com no és un sistema estrictament posicional, és complexe implementar les operacions, resultant en unes operacions molt més lentes.

Decimal	Binari	Decimal	Binari
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Codi BCD per a números

Codi BCD (Binary coded decimal). És molt útil per representar números en displays (pantalles), per fer operacions fent servir números en punt fix (operacions monetàries).

Cada dígit decimal té una representació binària codificada amb **4 bits**.

Inconvenients:

- Amb el mateix número de bits es representen menys números que amb el binari natural.
- Ja no fem servir un sistema posicional pur; les regles canvien i les operacions són més complexes.

Decimal	Binari	Decimal	Binari
0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

Exemple: 59237

Decimal: 5 · 9 · 2 · 3 · 7

BCD: 0101 · 1001 · 0010 · 0011 · 0111

Codificació ASCII per dades alfanumèriques

Codi ASCI. Serveix per a la representació de caràcters alfanumèrics, de puntuació, de control, símbols matemàtics i tota mena de caràcters no numèrics. Els 31 primers caràcters són caràcters de control, i fins al 127 representen els dígits decimals, les lletres majúscules i minúscules del anglès, els caràcters usuals de puntuació.

Així els 128 primers caràcters (des de zero fins a 127) és el codi ASCII estàndard (7 bits), i es van reservar 128 caràcters addicionals per a símbols d'altres idiomes i símbols gràfics. En total 256 caràcters, representats en 8 bits (1 byte).

Desgraciadament 128 caràcters eren clarament insuficients per codificar tots els símbols necessaris per a tots els idiomes del mon, ni fent servir diferents codificacions per a cada idioma incompatibles entre elles.

ASCII	Hex	Simbolo
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Simbolo
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

ASCII	Hex	Simbolo
32	20	(espacio)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Simbolo
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Simbolo
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Simbolo
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Simbolo
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Simbolo
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	□

Unicode i UTF: Ampliació del codi ASCII

Unicode és un estàndard internacional de codificació de caràcters, per a suports informàtics. Permet emmagatzemar qualsevol mena d'escriptura que es faci servir actualment, moltes formes d'escriptura conegudes només pels estudiosos, i símbols com ara els símbols matemàtics, lingüístics, i APL.

- Arab
- Cil·liric
- Grec
- Braille
- Taquigrafia
- Japonès
- Xinès (simplificat)
- …

Fa servir 32 bits, és a dir, 4 bytes.

- **UTF-8**: 8 bits, amb símbols de longitud variable (1,2,4 bytes)
- **UTF-16**: 16 bits de longitud variable, optimitzada al pla bàsic multilingüe (BMP) que conté la gran majoria de caràcters i sistemes d'escriptura en ús en l'actualitat. codificació en 2, 4 bytes De vegades es limita UTF-16 a 16 bits de longitud fixe.
- **UTF-32**: 32 bits de longitud fixa, la més senzilla de les tres. Directament Unicode (4 bytes).

UTF és **compatible amb ASCII en els 128 primers caràcters**, obtenint compatibilitat enrere amb l'anglès. UTF-8 fa servir 1 byte per aquests caràcters.

Els textos en UTF poden incorporar un prefix inicial amb uns bytes indicant la codificació, i l'ordre dels bytes en cada paraula (***BOM**: Byte Order Mark*).

Números en punt flotant

En el punt fix la precisió és independent de la magnitud del número, però un error de 10cm en una magnitud de 1m no és el mateix que un error de 10cm en una medició de 300Km.

En el punt flotant la precisió, i per tant l'error, és relativa a la magnitud del número:

1.32478123E+35
Error màxim: 0.000000004E+35 = 4.0E+26
Error relatiu: 4.0E+26 / 1.32478123E+35 = **3.02E-9**

1.32478123E+01
Error màxim: 0.000000004E+01 = 4.0E-8
Error relatiu: 4.0E-8 / 1.32478123E+01 = **3.02E-9**

El punt flotant consta de **signe**, **mantissa** i **exponent**.

L'**estàndard de la IEEE per l'aritmètica en coma flotant** defineix:

- **precisió simple** (**32 bits**): 1 bit de signe, 8 d'exponent i 23 de mantissa.
- **precisió doble** (**64 bits**): 1 bit de signe, 11 d'exponent i 52 de mantissa.

A més de la representació de números inclou els valors especials de ***∞*** i ***-∞***, el valor ***NaN*** (***N****ot a* ***N****umber*, per indicar operancions fora de domini: 0/0, ∞/∞, arrel quadrada d'un número negariu…)

Numeros decimals. index

Representacions:

1. Punt fix en binari
2. Punt flotant
   a. Precisió simple (7 xifres)
3. Precisió doble (15 xifres)
4. Punt fix/flotant en BCD
   a. COBOL
   b. Llibreries específiques
      - Python *Decimal*

```
float 4 bytes (32 bits)
3.4e-038..3.4e+038

double 8 bytes (64 bits)
1.7e-308..1.7e+308
```

- Les codificacions en binari no permeten representacions de tots els números decimals; pe. 0.1 en binari és periòdic.
  - Apareixen errors de càlcul.
- Existeixen codificacions BCD en cadenes de longitud variable, però totes les representacions en un nombre fix de bits incorporen una limitació (***overflow***, tant pels números positius com negatius).
- L'error en els números en punt flotant és relatiu a la magnitud.
- Permet la representació dels valors especials que es propaguen en les expressions:
  - ***∞*** i ***-∞***
  - ***NaN***
  - També permet "retardar" el cas d'***underflow*** amb pèrdua de precisió.