

Representació de la informació

A. Josep Velasco González

Amb la col·laboració de:
Ramon Costa Castelló
Montse Peiron Guàrdia

PID_00215617

Índex

Introducció	5
Objectius	7
1. Els nombres i els sistemes de representació	9
1.1. Sistemes de representació	9
1.2. Sistemes de numeració posicionals	10
1.3. Canvis de base	12
1.3.1. Mètode basat en el TFN	13
1.3.2. Mètode basat en el teorema de la divisió entera	14
1.3.3. Canvi de base entre b i b^n	16
1.4. Empaquetament de la informació	18
1.5. Nombres amb signe	21
1.6. Suma en els sistemes posicionals	21
1.7. Resta en els sistemes posicionals	22
1.8. Multiplicació i divisió per potències de la base de numeració	23
2. Representació dels nombres en un computador	26
2.1. Condicionants físics	26
2.1.1. Rang de representació	27
2.1.2. Precisió	28
2.1.3. Error de representació	28
2.1.4. Aproximacions: truncament i arrodoniment	28
2.1.5. Sobreiximent	30
2.2. Nombres naturals	31
2.3. Nombres enters	33
2.3.1. Representació d'enters en signe i magnitud en base 2	33
2.3.2. Suma i resta en signe i magnitud	35
2.3.3. Representació en complement a 2	36
2.3.4. Canvi de signe en complement 2	38
2.3.5. Magnitud dels nombres en complement a 2	40
2.3.6. Suma en complement a 2	40
2.3.7. Resta en complement a 2	42
2.3.8. Multiplicació per 2^k de nombres en complement a 2	43
2.4. Nombres fraccionaris	44
3. Altres tipus de representacions	53
3.1. Representació d'informació alfanumèrica	53
3.2. Codificació de senyals analògics	55
3.3. Altres representacions numèriques	58
3.3.1. Representació en excés a M	58

3.3.2. Representació en coma flotant	59
3.3.3. Representació BCD	64
Resum	66
Exercicis d'autoavaluació	67
Solucionari	68
Glossari	94
Bibliografia	95

Introducció

Inicialment, els computadors es van desenvolupar com una eina per a agilitar la repetició d'operacions aritmètiques i lògiques bàsiques que amb el temps van anar guanyant complexitat, tant pel nombre d'operacions com per la complexitat pròpia dels càlculs. Avui dia, sense haver perdut la utilitat original, els computadors s'han anat diversificant i s'han adaptat a múltiples aplicacions fins a convertir-se en un element imprescindible en tots els camps de la ciència, de la comunicació i de l'oci.

Malgrat els grans canvis que han experimentat les màquines gradualment, el processament de les dades dins d'un computador continua basat en la realització d'operacions aritmètiques i lògiques senzilles sobre dades que es troben a la memòria principal. Allà poden haver arribat de procedències diverses, però en tots els casos, la informació ha experimentat una transformació: s'ha codificat de manera adequada per a poder tractar-la amb un processador digital.

Les característiques de la tecnologia amb la qual es construeixen els computadors obliguen a treballar amb només dos símbols diferents: el 0 i el 1. Tota la informació que hagi de processar un computador s'haurà de codificar fent servir únicament aquests dos símbols.

Dins d'un computador, qualsevol informació (valor numèric, text, àudio, vídeo) està representada com una cadena de 0 i 1. Ara bé, una cadena de 0 i 1 només té sentit si coneixem el format de representació, és a dir, la manera com està codificada la informació, la qual cosa inclou saber: el tipus de dada (és un nombre, un text, un senyal d'àudio digitalitzat, etc.) i el sistema utilitzat per a representar aquest tipus de dades (és a dir, el sistema de numeració, si és un nombre; la taula de codificació dels caràcters, si es tracta d'un text; l'algorisme de codificació o compressió per a informació multimèdia, etc.).

Nota

Es fan servir els símbols 0 i 1, perquè són els dígit binaris, el sistema que empren els computadors. A més, també es fan servir per a designar els termes *veritat* i *fals* en les operacions lògiques.

Què codifica la cadena 10100100? Doncs depèn, és clar. De quin tipus de dada es tracta? Si és un text i s'ha fet servir el codi ASCII ISO-8859-15, es tracta del símbol €; si és un nombre natural, es tracta del valor decimal 164; si és un enter codificat en signe i magnitud, és el valor decimal -36; si és un enter codificat en el sistema de complement a 2, és el valor decimal -92, etc. En tots els casos es tracta de la mateixa cadena, però en cada cas es considera que aquesta cadena és el resultat de codificar la informació d'una manera diferent.

Nota

Signe i magnitud i complement a 2 són sistemes de representació de nombres amb signe que es descriuen en la segona secció d'aquest mòdul.

La informació que processa un computador digital està codificada en cadenes de 0 i 1 i això vol dir que les operacions que es fan al computador són operacions sobre cadenes de 0 i 1. De fet, tot el processament que es fa als compu-

tadors es redueix a operacions aritmètiques i lògiques senzilles sobre les cadenes que codifiquen la informació.

Aquests són, doncs, els punts de partida:

- Dins d'un computador tota la informació es codifica com a cadenes de 0 i 1.
- Una cadena de 0 i 1 no té sentit per ella mateixa. Cal saber com es codifica la informació, ja que això és el format en el qual estan codificades les dades.
- El processament que duu a terme un computador sobre les cadenes de 0 i 1 consisteix en operacions aritmètiques i lògiques senzilles.

Majoritàriament, la informació dins dels computadores es tracta com a nombres i s'opera com a tal, per tant, conèixer la manera en la qual es codifiquen els nombres és bàsic per a entendre el funcionament dels computadores.

En aquest mòdul expliquem els sistemes bàsics de codificació de la informació, parant especial atenció a la representació de la informació numèrica, a la qual dediquem la major part del mòdul. El mòdul s'estructura de la manera següent. En primer lloc, es fa una anàlisi del sistema de numeració amb el qual estem habituats a treballar. A continuació, s'expliquen els sistemes de codificació de nombres més usats als computadores i, finalment, es donen les pautes per a la codificació de dades no numèriques.

Objectius

A continuació enumerem els objectius principals que cal assolir amb l'estudi d'aquest mòdul:

1. Comprendre com es pot representar qualsevol tipus d'informació dins dels computadors i aprendre els principis bàsics de la codificació.
2. Aprendre en profunditat els sistemes ponderats no redundants de base fixa 2, 10 i 16, a més de saber representar un mateix valor numèric en bases diferents.
3. Comprendre i saber fer servir els formats amb què es codifica la informació numèrica en un computador: el sistema ponderat en binari per als nombres naturals; signe i magnitud i complement a 2 per als nombres enters, i la representació de nombres fraccionaris en coma fixa.
4. Aprendre les operacions aritmètiques bàsiques que duu a terme un computador i saber fer-les a mà. Aquestes operacions són la suma, la resta i la multiplicació i la divisió entre potències de la base de nombres naturals, enters i fraccionaris.
5. Comprendre els conceptes de rang i precisió d'un format de codificació de la informació numèrica en un computador, i també els conceptes de sobreiximent i d'error de representació.
6. Entendre la manera d'empaquetar cadenes d'uns i zeros a partir de la base 16.
7. Aprendre la manera de representar caràcters en format ASCII.

1. Els nombres i els sistemes de representació

L'objectiu d'aquesta secció és analitzar el sistema de numeració que fem servir i identificar els paràmetres que el defineixen. Per fer-ho, introduïrem els conceptes d'arrel o base i de pes associat a la posició d'un dígit. Seguidament, explicarem les tècniques per trobar la representació d'un nombre en un sistema posicional d'arrel fixa quan es canvia l'arrel. Finalment, farem una anàlisi de l'operació més comuna, la suma, i de la seva homòloga, la resta, i també de la multiplicació i de la divisió de nombres per potències de la base de numeració.


Terminologia

Al llarg del text farem servir indistintament els termes **representar** i **codificar** per a referir-nos a la manera com s'escriu una dada segons una sintaxi i un conjunt de símbols determinats.

1.1. Sistemes de representació

La idea de valor numèric és un concepte abstracte que determina una quantitat. Els valors numèrics estan subjectes a un ordre de precedència que es fa servir per a relacionar-los i arribar a conclusions. Per tal de treballar de manera àgil amb aquest tipus d'informació, hem de poder representar els valors numèrics de manera eficient, per la qual cosa s'han desenvolupat els anomenats **sistemes de numeració**.

Un **sistema de numeració** és una metodologia que permet representar un conjunt de valors numèrics.

El ventall de sistemes de numeració és bastant ampli. Entre altres, podem trobar els sistemes d'arrel o base, els sistemes de díigits signats, els sistemes de residus i els sistemes racionals. D'aquests, els sistemes basats en arrel (o base) són els que més es fan servir pels avantatges que aporten en la manipulació aritmètica dels valors numèrics, i en ells centrarem la nostra atenció. 

Terminologia

Es pot utilitzar la designació de *base* o *arrel* de manera indistinta, tot i que és més usual l'ús del mot *base*: parlem de sistemes de numeració en *base n*.

Un **sistema de numeració basat en arrel** descriu els valors numèrics en funció d'una o diverses arrels. L'arrel o base del sistema de numeració indica el nombre de díigits diferents de què disposem.

Quan treballem amb base 10, disposem de deu símbols diferents, que anomenem **díigits**, per a la representació: 0, 1, 2, 3, 4, 5, 6, 7, 8 i 9. Si la base del sistema de numeració és 2, disposem de dos díigits, el 0 i l'1. En base 16, hi ha setze díigits diferents, que es representen amb: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F.

Terminologia

Dígit: cada un dels signes gràfics emprats per a representar els nombres en un sistema de numeració.

Els sistemes de numeració que fan servir només una base reben el nom de **sistemes de numeració de base fixa**. El sistema de numeració que fem servir en la nostra aritmètica quotidiana és un sistema de numeració de base fixa en el qual la base de numeració és 10.

Considerem el nombre 321 en el nostre sistema de numeració en base 10. Hem fet servir el dígit 3, el dígit 2 i el dígit 1, ordenats d'una manera determinada. Aquests mateixos dígitos ordenats d'una altra manera (per exemple, 213) representen un nombre diferent, tot i que està constituït pels mateixos dígitos. Els sistemes de numeració en els quals l'ordre dels dígitos és determinant en la representació numèrica s'anomenen **sistemes posicionals**.

Un **sistema de numeració posicional** és aquell en què la representació d'un valor numèric està determinada per una seqüència **ordenada** de dígitos.

A partir d'aquest punt, les anàlisis i els estudis continguts a la resta d'apartats d'aquest mòdul fan referència a sistemes de numeració posicionals de base fixa, que són els que tenen més interès per a l'estudi de la representació de la informació numèrica en els computadors. !

1.2. Sistemes de numeració posicionals

Entenem que el 632 en base 10 representa 6 centenes, 3 desenes i 2 unitats. És a dir, els dígitos tenen pes 100, 10 i 1, respectivament. Un canvi d'ordre dels dígitos (per exemple, 326), canvia els pesos associats a cada dígit i, per tant, el nombre representat. En un sistema de numeració posicional, cada dígit té associat un pes que depèn de la posició i de la base de numeració. !

Un **sistema de numeració posicional de base fixa** és aquell en què un valor numèric X es representa com una seqüència **ordenada** de dígitos, de la manera següent:

$$x_{n-1}x_{n-2} \cdots x_1x_0, x_{-1} \cdots x_{-m}$$

en la qual cada x_i és un dígit de manera que $0 \leq x_i \leq b-1$, en la qual b és la base del sistema de numeració i x_i el dígit de la posició i -èsima de la seqüència.

Les posicions amb subíndex negatiu corresponen a la **part fraccionària** del nombre, mentre que les posicions amb subíndex positiu corresponen a la **part entera**. La frontera entre la part entera i la part fraccionària s'indica amb una **coma**. Els dígitos de la part entera es consignen a l'esquerra de la coma i els de la part fraccionària a la dreta de la coma.

Sistemes de numeració de base mixta

Són els que fan servir més d'una base de numeració. Un exemple d'aquest tipus de sistema és el sistema horari, en el qual els valors vénen donats en funció de les bases 24, 60 i 60 (hores, minuts i segons).

Terminologia

Farem servir X per a referir-nos al concepte abstracte de valor numèric. La representació del valor numèric X en base b l'escriurem de la forma $X_{(b)}$, en la qual b és la base en decimal.

El sistema de numeració de base 10 amb què treballem habitualment rep el nom de **sistema decimal**. De manera anàloga, anomenem **sistema hexadecimal** el sistema de numeració en base 16, **sistema octal** el que fa servir base 8 i **sistema binari** el que fa servir base 2. Els dígit binaris reben el nom de **bits**. !

Considerem, de nou, el nombre $632_{(10)}$. El podem escriure en funció dels pesos associats a cada posició:

$$632_{(10)} = 6 \cdot 100 + 3 \cdot 10 + 2 \cdot 1$$

Segons la definició, el 2 ocupa la posició 0, el 3 la posició 1 i el 6 la posició 2. Podem reescriure l'expressió anterior relacionant els pesos amb la base de numeració i amb la posició que ocupa cada dígit:

$$632_{(10)} = 6 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0$$

El $34,75_{(10)}$ també es pot escriure en funció de la base i de la posició de cada dígit:

$$34,75_{(10)} = 3 \cdot 10^1 + 4 \cdot 10^0 + 7 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

En general, un sistema de representació numèrica posicional de base fixa permet expressar un valor numèric en funció de la base de numeració i de la posició de cada dígit. !

La seqüència de dígit que representa un valor numèric en un sistema posicional ha de ser **ordenada** perquè cada posició té un pes associat. Aquest pes depèn de la posició i de la base de numeració. El pes associat a la posició p és b^p , en la qual b és la base de numeració. !

El nombre X representat per la seqüència de dígit $x_{n-1}x_{n-2} \cdots x_1x_0, x_{-1} \cdots x_{-m}$ es pot expressar en funció de la base de numeració de la manera següent:

$$X = \sum_{i=-m}^{n-1} x_i b^i = x_{n-1} \cdot b^{n-1} + x_{n-2} \cdot b^{n-2} + \cdots + x_{-m} \cdot b^{-m}$$

en la qual cada x_i és un dígit de manera que $0 \leq x_i \leq b-1$, en la qual b és la base del sistema de numeració i x_i el dígit de la posició i -èsima de la seqüència.

Terminologia

Evitem fer servir l'expressió *part decimal* per a designar la part fraccionària d'un nombre i així eludirem l'ambigüitat del terme *nombre decimal*. Un **nombre decimal** és un nombre en base 10, no un nombre amb part fraccionària.

Terminologia

Un dígit binari rep el nom de **bit**, que és un acrònim de l'expressió anglesa *binary digit*.

Nota

Segons la numeració de posicions definida, el 7 ocupa la posició -1 i el 5 la posició -2 , mentre que el 3 i el 4 (dígit de la part entera) ocupen les posicions 1 i 0, respectivament.

Recordem que $x^{-k} = \frac{1}{x^k}$


Terminologia

Expressar un nombre en funció de la base de numeració equival a escriure'l de la manera següent:

$$x_{n-1} \cdot b^{n-1} + x_{n-2} \cdot b^{n-2} + \cdots$$

Aquesta expressió es coneix com el **teorema fonamental de la numeració (TFN)***.

** Abreujarem **teorema fonamental de la numeració** amb la sigla **TFN**.*

D'aquest teorema es desprèn que, a més de la seqüència de dígit, en un sistema posicional d'arrel fixa cal conèixer la base de numeració per a determinar el valor numèric representat. 

La seqüència de dígit 235 és vàlida en totes les bases més grans que 5 (perquè el 5 no és un dígit vàlid en bases inferiors a 6). Ara bé, en bases diferents, representa nombres diferents. Per tant, $235_{(6)} \neq 235_{(10)} \neq 235_{(16)}$. La taula següent mostra la correspondència entre les representacions d'alguns valors numèrics en les bases més usuals:

Base 2	Base 4	Base 8	Base 10	Base 16
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	3	3	3	3
100	10	4	4	4
101	11	5	5	5
110	12	6	6	6
111	13	7	7	7
1000	20	10	8	8
1001	21	11	9	9
1010	22	12	10	A
1011	23	13	11	B
1100	30	14	12	C
1101	31	15	13	D
1110	32	16	14	E
1111	33	17	15	F
10000	100	20	16	10
10001	101	21	17	11
10010	102	22	18	12

Elements de la taula

En cada columna es representen els valors numèrics des del 0 fins al $18_{(10)}$ en la base indicada en la casella superior de la columna. En cada fila disposem la representació del mateix valor numèric en diferents bases.

1.3. Canvis de base

La seqüència ordenada de dígit que representa un valor numèric canvia segons la base del sistema de numeració, però hi ha una relació entre les seqüències de dígit.

Els **mètodes de canvi de base** permeten trobar la seqüència ordenada de dígit que representa un valor numèric X en el sistema de numeració en base b' , a partir de la representació en el sistema de numeració en base b , és a dir:

$$\text{canvi_a_base_}b'(X_{(b)}) = X_{(b')}$$

Ús dels canvis de base

Farem servir els canvis de base per a convertir la representació d'un nombre entre les bases 2, 10 i 16.


En els apartats següents exposem dues tècniques de canvi de base.

1.3.1. Mètode basat en el TFN

Si apliquem el TFN al $324_{(10)}$ el podem escriure com a:

$$324_{(10)} = 3 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0$$

Si fem les operacions de la dreta en base 10, obtenim la representació en base 10, que és la que tenim a l'esquerra de la igualtat. Ara bé, si fem les operacions en base 7, tindrem la representació en base 7. En general, si fem les operacions en base b obtenim la representació en base b .

Com que la dificultat està a operar en una base que no sigui base 10 (perquè no hi estem acostumats), el mètode serà útil per a passar a base 10. 

Canvi de base segons el TFN

Per canviar a base 10 el $462_{(7)}$:

1) Expressem el nombre en funció de la base (base 7) segons el TFN:

$$462_{(7)} = 4 \cdot 7^2 + 6 \cdot 7^1 + 2 \cdot 7^0$$

2) Fem les operacions en la base d'arribada (base 10):

$$4 \cdot 7^2 + 6 \cdot 7^1 + 2 \cdot 7^0 = 4 \cdot 49 + 6 \cdot 7 + 2 \cdot 1 = 240_{(10)}$$


Les seqüències de dígit $462_{(7)}$ i $240_{(10)}$ representen el mateix valor numèric, però en bases diferents: base 7 la primera i base 10 la segona.

Per trobar la representació de $X_{(b)}$ en base 10 hem de fer el següent:

- 1) Expressar $X_{(b)}$ en funció de la base b , seguint el TFN.
- 2) Fer les operacions en base 10.

Quan $b > 10$, els dígit de la base b s'han de canviar a base 10 abans de fer les operacions.

Valors decimals	Dígits hexadecimals
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

El mètode és vàlid tant per a nombres enters com per a nombres amb part fraccionària. 

Canvis de base segons el TFN

Per passar a base 10 el nombre $101100,01_{(2)}$:

1) Expressem el nombre en funció de la base (base 2):

$$101100,01_{(2)} = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

2) Fem les operacions en base 10:

$$\begin{aligned} 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = \\ 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0,5 + 1 \cdot 0,25 = 44,25_{(10)} \end{aligned}$$

El $101100,01_{(2)}$ en base 10 és el $44,25_{(10)}$.

Per passar a base 10 el nombre $AF2C,2_{(16)}$:

1) Expressem el nombre en funció de la base (base 16):

$$AF2C,2_{(16)} = A \cdot 16^3 + F \cdot 16^2 + 2 \cdot 16^1 + C \cdot 16^0 + 2 \cdot 16^{-1}$$

2) Fem les operacions en base 10. En aquest cas, hem de canviar a base 10 els dígit hexadecimals abans de fer les operacions:

$$\begin{aligned} A \cdot 16^3 + F \cdot 16^2 + 2 \cdot 16^1 + C \cdot 16^0 + 2 \cdot 16^{-1} = \\ 10 \cdot 16^3 + 15 \cdot 16^2 + 2 \cdot 16^1 + 12 \cdot 16^0 + 2 \cdot 16^{-1} = 44844,125_{(10)} \end{aligned}$$

El $AF2C,2_{(16)}$ en base 10 és el $44844,125_{(10)}$.

1.3.2. Mètode basat en el teorema de la divisió entera


Aquest mètode de canvi de base consisteix a fer divisions enteres per la nova base de numeració de manera iterativa. Els residus de les divisions enteres són els dígit de la representació en la nova base.


Per canviar a base 7 el nombre $317_{(10)}$, fem divisions enteres per 7:

$$\begin{array}{rcl} 317 & = & 45 \cdot 7 + 2 \\ 45 & = & 6 \cdot 7 + 3 \\ 6 & = & 0 \cdot 7 + 6 \end{array} \quad \uparrow$$

$$317_{(10)} = 632_{(7)}$$

La seqüència de residus en **ordre invers** ens dona la representació en la nova base. El nombre $317_{(10)}$ en base 7 és el $632_{(7)}$.

Com que les operacions es fan en la base inicial, aquest mètode és especialment útil per a passar de base 10 a una altra. 

Per tal de canviar de base de nombres fraccionaris amb aquest mètode, hem de tractar per **separat** la part entera i la part fraccionària. 

Per trobar la representació de $X_{(10)}$ en base b :

1) **Part entera**: successivament, hem de fer en base 10 la divisió entera per la nova base b . Aturem la successió de divisions quan obtenim un quocient 0. La seqüència de residus, agafats del darrer al primer, és la seqüència de dígit d'esquerra a dreta de la part entera en la nova base. Quan $b > 10$, els residus s'han de passar a dígit de la nova base.

Exemple

Un nombre amb part fraccionària finita no periòdica en una base pot tenir una part fraccionària infinita periòdica en una altra base. Per exemple,

$$0,3_{(10)} = 0,01001\overline{1001}_{(2)}.$$

2) Part fraccionària: successivament, se separa la part fraccionària i es multiplica per la nova base b . Les operacions es fan en base 10. Aturem la successió de multiplicacions quan trobem un comportament periòdic o quan tinguem dígitos suficients. La seqüència de valors enters obtinguts en fer les multiplicacions agafats del primer al darrer és la seqüència de dígitos d'esquerra a dreta en la nova base de representació. Quan $b > 10$, els enters obtinguts s'han de passar a dígitos de la nova base.

Finalment, cal ajuntar la part entera i la part fraccionària obtingudes.

Canvis de base pel mètode de la divisió entera

Canviem a base 2 el $44,25_{(10)}$:

a) Part entera: successivament, fem divisions enteres per la nova base (base 2) fins a obtenir un quocient 0 i prenem els residus en ordre invers:

$$\begin{array}{rcl} 44 & = & 22 \cdot 2 + 0 \\ 22 & = & 11 \cdot 2 + 0 \\ 11 & = & 5 \cdot 2 + 1 \\ 5 & = & 2 \cdot 2 + 1 \\ 2 & = & 1 \cdot 2 + 0 \\ 1 & = & 0 \cdot 2 + 1 \end{array} \quad \uparrow$$

$$44_{(10)} = 101100_{(2)}$$

b) Part fraccionària: successivament, multipliquem per la nova base (base 2):

$$\begin{array}{rcl} 0,25 \cdot 2 & = & 0,50 = 0,50 + 0 \\ 0,50 \cdot 2 & = & 1,00 = 0,00 + 1 \end{array} \quad \downarrow$$

$$0,25_{(10)} = 0,01_{(2)}$$

Per tal de completar el canvi de base, ajuntem la part entera i la part fraccionària que en resulten:

$$44,25_{(10)} = 101100_{(2)} + 0,01_{(2)} = 101100,01_{(2)}$$

El $44,25_{(10)}$ en base 2 és el $101100,01_{(2)}$.

Canviar a base 16 el $44844,12_{(10)}$:

a) Part entera: successivament, fem divisions enteres per 16 fins a obtenir un quocient 0:

$$\begin{array}{rcl} 44844 & = & 2802 \cdot 16 + 12 \\ 2802 & = & 175 \cdot 16 + 2 \\ 175 & = & 10 \cdot 16 + 15 \\ 10 & = & 0 \cdot 16 + 10 \end{array} \quad \uparrow$$

Com que la nova base és més gran que 10, hem de convertir els residus a la nova base (base 16):

$$\begin{array}{l} 12_{(10)} = C_{(16)} \\ 2_{(10)} = 2_{(16)} \\ 15_{(10)} = F_{(16)} \\ 10_{(10)} = A_{(16)} \end{array} \quad 44844_{(10)} = AF2C_{(16)}$$

b) Part fraccionària: successivament, multipliquem per 16:

$$\begin{array}{rcl} 0,12 \cdot 16 & = & 1,92 = 0,92 + 1 \\ 0,92 \cdot 16 & = & 14,72 = 0,72 + 14 \\ 0,72 \cdot 16 & = & 11,52 = 0,52 + 11 \\ 0,52 \cdot 16 & = & 8,32 = 0,32 + 8 \\ 0,32 \cdot 16 & = & 5,12 = 0,12 + 5 \\ 0,12 \cdot 16 & = & 1,92 = 0,92 + 1 \\ 0,92 \cdot 16 & = & 14,72 = 0,72 + 14 \\ & & \dots \end{array} \quad \downarrow$$

La seqüència d'enters que obtenim es repeteix (1,14,11,8,5,1,14,...). Per tant, és un nombre periòdic. A més, els enters s'han de convertir a dígits de base 16:

$$\begin{aligned}1_{(10)} &= 1_{(16)} \\14_{(10)} &= E_{(16)} \\11_{(10)} &= B_{(16)} \\8_{(10)} &= 8_{(16)} \\5_{(10)} &= 5_{(16)} \\1_{(10)} &= 1_{(16)}\end{aligned}$$

$$0,12_{(10)} = 0,1EB851EB851EB..._{(16)} = 0,\overline{1EB85}_{(16)}$$

Finalment, juntarem la part entera i la part fraccionària:

$$44844,12_{(10)} = AF2C,\overline{1EB85}_{(16)}$$

El $44844,12_{(10)}$ en base 16 és el $AF2C,\overline{1EB85}_{(16)}$.

Si hem de canviar de base b a base b' , i ni b ni b' són la base 10, fem servir la base 10 com a base intermèdia. Així, fem servir el primer mètode (basat en el TFN) per a passar de base b a base 10 i, posteriorment, el segon mètode (basat en el teorema de la divisió entera) per a passar de base 10 a base b' .

Canvi entre bases diferents de la base 10

El canvi a base 6 del $232,1_4$ l'hem de fer en dues passes:

1) Fem el canvi a base 10 del $232,1_4$ aplicant el mètode del TFN:

$$\begin{aligned}232,1_4 &= 2 \cdot 4^2 + 3 \cdot 4^1 + 2 \cdot 4^0 + 1 \cdot 4^{-1} = \\&= 32 + 12 + 2 + 0,25 = 46,25_{(10)}\end{aligned}$$

2) Fem el canvi a base 6 del $46,25_{(10)}$:

$$\begin{array}{rcl}46 &= 7 \cdot 6 + 4 & \uparrow \\7 &= 1 \cdot 6 + 1 & \\1 &= 0 \cdot 6 + 1 & \end{array} \quad \begin{array}{rcl}0,25 \cdot 6 &= 1,50 &= 0,50 + 1 \\0,50 \cdot 6 &= 3,00 &= 0,00 + 3 \quad \downarrow\end{array}$$

$$46_{(10)} = 114_6$$

$$0,25_{(10)} = 0,13_6$$

El $46,25_{(10)}$ és equivalent al $114,13_6$

Per tant, el $232,1_4$ és el $114,13_6$ en base 6.

1.3.3. Canvi de base entre b i b^n

El canvi de base b a base b^n és directe perquè...


... un dígit en base b^n es correspon amb n dígits en base b .

Potències de 2	
2^{16}	65536
2^{15}	32768
2^{14}	16384
2^{13}	8192
2^{12}	4096
2^{11}	2048
2^{10}	1024
2^9	512
2^8	256
2^7	128
2^6	64
2^5	32
2^4	16
2^3	8
2^2	4
2^1	2
2^0	1
2^{-1}	0,5
2^{-2}	0,25
2^{-3}	0,125
2^{-4}	0,0625
2^{-5}	0,03125
2^{-6}	0,015625
2^{-7}	0,0078125
2^{-8}	0,00390625

Aquesta circumstància es dona entre base 2 i base 16 (perquè $16 = 2^4$) o entre base 16 i base 4 (perquè $16 = 4^2$), però no entre base 8 i base 16, perquè 16 no és potència de 8.

Canvi de base b a base b^n

En el canvi a base 16 del $10010110,01101101_{(2)}$, tindrem en compte que 16 és potència de 2: $16 = 2^4$. Aquesta relació indica que cada dígit de base 16 es correspon amb quatre dígits de base 2.

El canvi de base s'aconsegueix si fem agrupacions de quatre dígits binaris i convertim cada agrupació en un dígit hexadecimal. Les agrupacions es fan sempre partint de la coma fraccionària i han de ser completes. Si manquen dígits per a completar una agrupació, afegirem zeros. 

Vegeu la correspondència entre binari i hexadecimal en la taula del subapartat 1.2.

$$\begin{array}{cccccc} 1001 & 0110 & , & 0110 & 1101 & (2 \\ 9 & 6 & , & 6 & D & (16 \end{array}$$

El $10010110,01101101_{(2)}$ és en base 16 el $96,6D_{(16)}$.

En el canvi a base 16 del $101110,101101_{(2)}$, hem de completar les agrupacions afegint zeros (en aquest cas, tant a la part entera com a la fraccionària):

$$\begin{array}{cccccc} 0010 & 1110 & , & 1011 & 0100 & (2 \\ 2 & E & , & B & 4 & (16 \end{array}$$

$$101110,101101_{(2)} = 2E,B4_{(16)}$$

Canvi de base b^n a base b

Quan el canvi és de base b^n a b , el procediment és anàleg però en sentit invers: cada dígit en base b^n es transforma en n dígits en base b .

Per canviar a base 2 el $7632,13_{(8)}$, tindrem en compte que $8 = 2^3$. Consegüentment, cada dígit en base 8 donarà lloc a tres dígits binaris:

$$\begin{array}{cccccc} 7 & 6 & 3 & 2 & , & 1 & 3 & (8 \\ 111 & 110 & 011 & 010 & , & 001 & 011 & (2 \end{array}$$

$$7632,13_{(8)} = 111110011010,001011_{(2)}$$

Hem de parar esment al fet que s'han d'obtenir exactament n dígits en base b per cada dígit en base b^n (en aquest cas, tres dígits binaris per cada dígit octal) afegint per cada dígit els zeros necessaris. Vegem com en el canvi a base 2 del $E1B2,4F_{(16)}$ cada dígit hexadecimal dona lloc a quatre dígits binaris.

$$\begin{array}{cccccc} E & 1 & B & 2 & , & 4 & F & (16 \\ 1110 & 0001 & 1011 & 0010 & , & 0100 & 1111 & (2 \end{array}$$

$$E1B2,4F_{(16)} = 1110000110110010,01001111_{(2)}$$

Errors freqüents

Sovint es cometen dos errors en aquests tipus de canvi de base:

1) Quan fem un canvi de base b'' a base b , cada dígit de base b'' ha de donar lloc, exactament, a n dígits en base b . Cal evitar l'error següent:

$$A3_{(16)} = 101011_{(2)}$$

en el qual el dígit A ha donat lloc als bits 1010 i el dígit 3 als bits 11. En realitat, ha de ser:

$$A3_{(16)} = 10100011_{(2)}$$

en el qual s'han afegit dos zeros per tal de completar el conjunt de quatre dígits que ha de generar el dígit hexadecimal 3.

2) Quan fem un canvi de la base b a la base b'' , són necessaris n dígits de la base b per a obtenir un dígit en la base b'' . Cal evitar l'error següent:

$$1100,11_{(2)} = C,3_{(16)}$$

en el qual els bits 1100 donen lloc al dígit hexadecimal C i els bits 11 al 3. En realitat, ha de ser:

$$1100,1100_{(2)} = C,C_{(16)}$$

en el qual s'han afegit 2 zeros per la dreta a fi de constituir un grup de quatre dígits binaris que donen lloc al dígit hexadecimal C.

1.4. Empaquetament de la informació


Amb els canvis a base 2 tenim un camí obert per a processar els nombres dins dels computadors. De fet, dins dels computadors, tota la informació (no només la numèrica) es codifica fent servir, únicament, la xifra 1 i la xifra 0. Per tant, tota la informació que processa un computador està codificada en cadenes d'1 i 0, és a dir, en cadenes de bits.

Ara bé, disposar només de dos símbols ens porta a representacions amb un nombre de dígits gran, a cadenes de bits llargues, que per nosaltres (no per als computadors) són difícils de recordar i de manipular.

Doncs bé, podem aprofitar la tècnica de fer agrupacions de quatre bits (vegeu els canvis de base 2 a base 16) per a convertir les cadenes de bits en dígits hexadecimals i compactar-les, així, en cadenes molt més curtes i manejables.

Aquest procés rep el nom d'**empaquetament hexadecimal**. El procés invers s'anomena **desempaquetament**.

L'**empaquetament hexadecimal** consisteix a compactar informació binària en cadenes de dígit hexadecimal.

Habitualment es col·loca la lletra *h* al final de la cadena de dígit, per a indicar que són hexadecimal. 

Empaquetament d'una cadena de bits

Per empaquetar la cadena de bits 110100100011, procedim de la manera següent:


- 1) Dividim la cadena 110100100011 de dreta a esquerra en grups de 4 bits:


1101 – 0010 – 0011

- 2) Codifiquem cada grup de 4 bits com un dígit hexadecimal:

1101 – 0010 – 0011
D – 2 – 3

Per tant, si fem l'empaquetament hexadecimal de la cadena de bits 110100100011, obtenim D23h.

L'empaquetament hexadecimal s'empra molt en diferents àmbits relacionats amb els computadors per facilitar el treball amb nombres, instruccions i adreces de memòria. Aquest tipus d'empaquetament s'empra en cadenes de bits, amb independència del sentit que tinguin els bits de la cadena. 

El procés invers, el desempaquetament, permet recuperar la cadena de bits original. En aquest cas, cada dígit hexadecimal dóna lloc a 4 bits. Així, el dígit hexadecimal 4 donaria lloc al grup de 4 bits 0100 i no al 100. 


Desempaquetament

Per desempaquetar la cadena D23h, convertim els dígit hexadecimal a base 2 fent servir 4 bits per a cada un:

D – 2 – 3
1101 – 0010 – 0011

Per tant, si desempaquetem la cadena de dígit hexadecimal D23h, obtenim la cadena de bits 110100100011.

És important diferenciar el concepte d'empaquetament hexadecimal de cadenes de bits del concepte del canvi de base 2 a base 16. Quan fem un canvi de base 2 a base 16 d'un nombre hem de tenir present la posició de la coma fraccionària, perquè busquem la representació del mateix nombre però en base 16. Per tant, les agrupacions de 4 bits es fan a partir de la coma fraccionària: cap a l'esquerra de la coma per a obtenir els dígit hexadecimal enters i cap a la dreta per a aconseguir els dígit hexadecimal fraccionaris. En canvi, en l'empaquetament hexadecimal no es té en compte el sentit de la informació codificada i els bits s'agrupen de 4 en 4 de dreta a esquerra independentment

del seu sentit. En aquest cas, el que obtenim finalment no és la representació del nombre en base 16, sinó una cadena de dígit hexadecimals que codifiquen una cadena de bits. 

Vegem aquesta diferència segons si fem el canvi a base 16 del nombre $111010,11_{(2)}$ o l'empaquetament hexadecimal. Si volem fer el canvi a base 16, hem de fer agrupacions a partir de la coma fraccionària afegint els zeros necessaris per a completar les agrupacions tant per la dreta com per l'esquerra:

$$\begin{array}{ccccccc} 0011 & 1010 & , & 1100 & & &_{(2)} \\ 3 & A & , & C & & &_{(16)} \end{array}$$

En aquest cas, el resultat que obtenim indica que el nombre $111010,11_{(2)}$ en base 16 és el $3A,C_{(16)}$.

En canvi, si volem fer un empaquetament hexadecimal, les agrupacions es fan de dreta a esquerra, sense tenir en compte la posició de la coma. Es tracta com una tira d'1 i 0. El resultat final no guarda informació sobre la coma fraccionària:

$$\begin{array}{ccc} 1110 & 10,11 &_{(2)} \\ E & & B \end{array}$$

En aquest segon cas, el resultat que obtenim indica que l'empaquetament hexadecimal de la cadena de bits 11101011 és EBh. Podem comprovar que la seqüència de dígit hexadecimals que obtenim en un i altre cas pot ser diferent.

Activitats

1. Convertiu a base 10 els valors següents:

- a) $10011101_{(2)}$
- b) $3AD_{(16)}$
- c) $333_{(4)}$
- d) $333_{(8)}$
- e) $B2,3_{(16)}$
- f) $3245_{(8)}$
- g) $AC3C_{(16)}$
- h) $1010,11_{(8)}$
- i) $110011,11_{(4)}$
- j) $10011001,1101_{(2)}$
- k) $1110100,01101_{(2)}$

2. Convertiu a base 2 els valors següents:

- a) $425_{(10)}$
- b) $344_{(10)}$
- c) $31,125_{(10)}$
- d) $4365,14_{(10)}$

3. Convertiu a hexadecimal els nombres següents:

- a) $111010011,1110100111_{(2)}$
- b) $0,1101101_{(2)}$
- c) $111011,1010010101_{(2)}$
- d) $45367_{(10)}$

4. Convertiu els nombres hexadecimal següents a base 2, base 4 i base 8:

- a) $ABCD_{(16)}$
- b) $45,45_{(16)}$
- c) $96FF,FF_{(16)}$

5. Empleneu la taula següent:

Binari	Octal	Hexadecimal	Decimal
1101100,110			
	362,23		
		A1,03	
			74,3

En cada fila trobeu un valor numèric expressat en la base que indica la casella superior de la columna on es troba. Consigneu a la resta de caselles la representació corresponent segons la base indicada en la part superior.

6. Empaqueu en hexadecimal la cadena de bits 10110001.

7. Empaqueu en hexadecimal el nombre $0100000111,111010_{(2)}$ que està en un format de coma fixa de 16 bits, dels quals 6 són fraccionaris.

8. Desempaqueteu la cadena de bits A83h i:

- a) Trobeu el valor decimal si es tracta d'un nombre natural.
- b) Trobeu el valor decimal si es tracta d'un nombre en coma fixa sense signe de 12 bits, en el qual 4 són fraccionaris.

9. Considerem el nombre $1010,101_{(2)}$.

- a) Feu el canvi a base 16.
- b) Feu l'empaquetament hexadecimal.

1.5. Nombres amb signe

Quan representem magnituds, sovint els assignem un signe (+/-) que precedeix la magnitud i que indica si la magnitud és positiva o negativa. El símbol - identifica les magnituds negatives i el símbol + les positives:

$$\begin{array}{ll} +23_{(10)} & -456_{(8)} \\ -34,5_{(7)} & +AF,34_{(16)} \end{array}$$

Signe (+/-)

De vegades, quan treballem amb nombres amb signe, el signe positiu (+) no s'escriu i només apareix el signe quan es tracta d'un nombre negatiu.

Designarem els nombres que porten la informació de signe com a nombres amb signe en contraposició als nombres sense signe que només ens donen informació sobre la magnitud del valor numèric.

1.6. Suma en els sistemes posicionals

L'algorisme de suma de dos nombres decimals que estem habituats a fer servir progressa de dreta a esquerra i suma a cada etapa els dígit del mateix pes (els que ocupen la mateixa posició). Si la suma d'aquests dígit arriba al valor de la

base (10 en aquest cas), genera un **ròssec** (el que ens “emportem”) que se sumará amb els dígit de l’etapa següent:

$$\begin{array}{r}
 1 \quad 1 \quad \quad \quad \leftarrow \text{dígit d'arrossegament o ròssec*} \\
 8 \ 3 \ 4 \ 1 \ (10) \\
 + \quad 2 \ 4 \ 6 \ 3 \ (10) \\
 \hline
 1 \ 0 \ 8 \ 0 \ 4 \ (10) \leftarrow \text{resultat}
 \end{array}$$

* El dígit d'arrossegament o ròssec rep en anglès el nom de *carry*. Aquest terme és d'ús habitual dins l'entorn dels computadors.


En hexadecimal, se segueixen les mateixes pautes de suma, però tenint en compte que hi ha 16 dígit diferents:

$$\begin{array}{r}
 1 \quad \quad \quad \leftarrow \text{ròssecs} \\
 3 \ 5 \ 8 \ 2 \ (16) \\
 + \quad A \ F \ 1 \ 8 \ (16) \\
 \hline
 E \ 4 \ 9 \ A \ (16) \leftarrow \text{resultat}
 \end{array}$$

El procés de suma en base 2 és anàleg:

$$\begin{array}{r}
 1 \ 1 \ 1 \ 1 \ 1 \ 1 \quad \quad \quad \leftarrow \text{ròssecs} \\
 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ (2) \\
 + \quad 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ (2) \\
 \hline
 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ (2) \leftarrow \text{resultat}
 \end{array}$$

Taula de suma en base 2 ròssec / bit de suma		
+	0	1
0	0/0	0/1
1	0/1	1/0

Quan es produeix un ròssec a la darrera etapa de suma, el resultat té un dígit més que els sumands. 

1.7. Resta en els sistemes posicionals


L'operació de resta també es duu a terme de dreta a esquerra, s'operen els dígit d'igual pes i es considera el ròssec* de l'etapa precedent. La particularitat en aquesta operació és que el nombre de magnitud més petita (subtrahend) és el que s'ha de restar del nombre de magnitud més gran (minuend):

$$\begin{array}{r}
 8 \ 3 \ 4 \ 1 \ (10) \leftarrow \text{minuend} \\
 1 \ 1 \ 1 \quad \quad \quad \leftarrow \text{ròssecs} \\
 - \quad 2 \ 4 \ 6 \ 3 \ (10) \leftarrow \text{subtrahend} \\
 \hline
 5 \ 8 \ 7 \ 8 \ (10) \leftarrow \text{resultat}
 \end{array}$$

* En anglès, el ròssec, en el cas de la resta, rep el nom de *borrow*.

El procediment en altres bases és idèntic. Només cal adequar-se a la nova base i parar atenció de restar la magnitud petita de la gran:

$$\begin{array}{r}
 \text{A F 1 8}_{(16)} \\
 \quad 1 \quad \leftarrow \text{ròssecs} \\
 - \quad 3 \quad 5 \quad 8 \quad 2 \quad_{(16)} \\
 \hline
 \quad 7 \quad 9 \quad 9 \quad 6 \quad_{(16)} \leftarrow \text{resultat}
 \end{array}
 \qquad
 \begin{array}{r}
 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad_{(2)} \\
 \quad 1 \quad 1 \quad 1 \quad \quad \quad \leftarrow \text{ròssecs} \\
 - \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad_{(2)} \\
 \hline
 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad_{(2)} \leftarrow \text{resultat}
 \end{array}$$

En el cas de l'operació de resta no es pot produir cap ròssec a la darrera etapa. Per aquest motiu, el resultat d'una resta de nombres necessita, com a màxim, els mateixos dígit que el minuend. 

Taula de resta en base 2 ròssec / bit de resta			
		Minuend	
		0	1
Subtrahend	0	0/0	0/1
	1	1/1	0/0


1.8. Multiplicació i divisió per potències de la base de numeració


En un sistema posicional de base fixa, cada dígit té un pes b^p , en el qual b és la base de numeració i p la posició que ocupa el dígit. Els pesos associats als dígit dels nombres decimals són potències de 10. Consegüentment, multiplicar per 10 es tradueix en augmentar en una unitat la potència de 10 associada a cada dígit i dividir per 10 és equivalent a disminuir en una unitat la potència de 10 associada a cada dígit.

Segons el TFN, el nombre $56,34_{(10)} = 5 \cdot 10^1 + 6 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}$. Si multipliquem per 10 tenim:

$$\begin{aligned}
 56,34_{(10)} \cdot 10 &= (5 \cdot 10^1 + 6 \cdot 10^0 + 3 \cdot 10^{-1} + 4 \cdot 10^{-2}) \cdot 10 = \\
 &= 5 \cdot 10^2 + 6 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} = 563,4_{(10)}.
 \end{aligned}$$

L'efecte que s'obté és el desplaçament de la coma fraccionària. Multiplicar per 10 un nombre en base 10 és equivalent a desplaçar la coma fraccionària una posició a la dreta, mentre que dividir-lo per 10 és equivalent a desplaçar la coma una posició a l'esquerra. El procés es pot estendre a la multiplicació i divisió per una potència de 10: multiplicar per 10^k un nombre en base 10 equival a desplaçar la coma fraccionària k posicions a la dreta, i dividir-lo per 10^k equival a desplaçar la coma fraccionària k posicions a l'esquerra.

Aquest procés de multiplicació i divisió per potències de la base de numeració és vàlid en tots els sistemes de posicionals de base fixa b . 

 Consulteu els sistemes de numeració posicionals del subaparat 1.2 d'aquest mòdul.

Multiplicar per b^k un nombre en un sistema posicional de base fixa b equival a desplaçar la coma fraccionària k posicions a la dreta.

Dividir per b^k un nombre en un sistema posicional de base fixa b equival a desplaçar la coma fraccionària k posicions a l'esquerra.

Nombres sense part fraccionària

En un nombre sense part fraccionària, desplaçar la coma k posicions a la dreta equival a afegir k zeros a la dreta, atès que la part fraccionària és zero.

Multiplicació per una potència de 2 en binari

El resultat de multiplicar el $11010_{(2)}$ per 2^4 l'aconseguim si desplaçem la coma fraccionària 4 posicions a la dreta:

$$11010_{(2)} \cdot 2^4 = 110100000_{(2)}$$

Aquest resultat que obtenim de manera directa es pot justificar amb els càlculs següents:

$$\begin{aligned} 11010_{(2)} \cdot 2^4 &= (1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0) \cdot 2^4 = \\ &= (1 \cdot 2^8 + 1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4) = 110100000_{(2)} \end{aligned}$$

Per tant, $11010_{(2)} \cdot 2^4 = 110100000_{(2)}$.

Divisió per una potència de 2 en binari

El resultat de dividir el $11100_{(2)}$ per 2^2 l'aconseguim si desplaçem la coma fraccionària 2 posicions a l'esquerra:

$$11100_{(2)} / 2^2 = 111_{(2)}$$

Aquest resultat que obtenim de manera directa es pot justificar amb els càlculs següents:

$$\begin{aligned} 11100_{(2)} / 2^2 &= (1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0) / 2^2 = \\ &= (1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2}) = 111_{(2)} \end{aligned}$$

Per tant, $11100_{(2)} / 2^2 = 111_{(2)}$.

La divisió per una potència de la base de numeració d'un nombre sense part fraccionària pot donar com a resultat un nombre amb part fraccionària: $11100_{(2)} / 2^4 = 1,11_{(2)}$. Ara bé, podem donar el resultat en forma de dos nombres enters que reben el nom de **quocient i residu**: on el quocient té relació directa amb la part entera del resultat i el residu amb la part fraccionària. En aquest cas, l'operació rep el nom de **divisió entera**, mentre que, per oposició, la primera rep el nom de **divisió real**.

El quocient i el residu de la divisió entera de $11100_{(2)}$ per 2^4 s'obtenen a partir del resultat de la divisió real $11100_{(2)} / 2^4 = 1,11_{(2)}$, identificant el quocient amb la part entera (en aquest cas, el quocient és $1_{(2)}$) i el residu amb la part fraccionària multiplicada pel divisor (en aquest cas, el residu és $0,11_{(2)} \cdot 2^4 = 1100_{(2)}$).

El **quocient** i el **residu** d'una divisió entera d'un nombre enter per una potència de la base de numeració es poden obtenir a partir del resultat de divisió real, identificant el quocient amb la part entera i el residu amb la part fraccionària multiplicada pel divisor.

Activitats

10. Feu les operacions següents en la base especificada:

- a) $111011010_{(2)} + 100110100_{(2)}$
- b) $2345_{(8)} + 321_{(8)}$
- c) $A23F_{(16)} + 54A3_{(16)}$
- d) $111011010_{(2)} - 100110100_{(2)}$
- e) $2345_{(8)} - 321_{(8)}$
- f) $A23F_{(16)} - 54A3_{(16)}$

11. Feu les operacions següents en la base especificada:

- a) $62,48_{(16)} + 35,DF_{(16)}$

- b) $111101101,11011_{(2)} + 100110100,111_{(2)}$
- c) $62,48_{(16)} - 35,DF_{(16)}$
- d) $111101101,11011_{(2)} - 100110100,111_{(2)}$

12. Feu les multiplicacions següents:

- a) $128,7_{(10)} \cdot 10^4$
- b) $AFD_{(16)} \cdot 16^2$
- c) $1101,01_{(2)} \cdot 2^2$

13. Trobeu el quocient i el residu de les divisions enters següents:

- a) $52978_{(10)} / 10^3$
- b) $3456_{(16)} / 16^2$
- c) $100101001001_{(2)} / 2^8$

2. Representació dels nombres en un computador

En aquesta segona secció descrivim sistemes per representar nombres que s'usen per a codificar informació numèrica dins dels computadores.

2.1. Condicionants físics

Tot i les contínues millores tecnològiques, la capacitat d'emmagatzemament dels computadores és finita. Aquest fet condiciona la representació numèrica dins dels computadores, sobretot en nombres amb una part fraccionària infinita, com ara els casos ben coneguts dels nombres π o e , en general, en la representació de nombres irracionals, com $\sqrt{2}$.

Aquestes limitacions són semblants a les que trobem quan treballem amb llapis i paper. En els càlculs fets a mà fem servir el $3,14_{(10)}$ o el $3,1416_{(10)}$ com a aproximació a π . Dins dels computadores també es treballa amb aproximacions dels nombres que no es poden representar de manera exacta.

Quan un nombre no es pot representar de manera exacta dins d'un computador, es comet un **error de representació**. Aquest error és la distància entre el nombre que volem representar i el nombre representat realment.

Si representem el nombre π amb el valor $3,14_{(10)}$, estem cometent un error igual a $|\pi - 3,14_{(10)}| = 0,00159\dots$, mentre que si treballem amb el valor $3,1416_{(10)}$, l'error és $|\pi - 3,1416_{(10)}| = 7,3464102\dots \cdot 10^{-6}$.

Quan escrivim nombres ho fem de la manera més pràctica i adient en cada cas. Podem escriure 03, 3,00, 3,000 o simplement 3. En canvi, dins dels computadores cal seguir una pauta més rígida, un **format** que especifiqui i fixi el nombre de dígitos enters i fraccionaris amb els quals treballem. Si suposem un format de representació de la forma $x_2x_1x_0,x_{-1}x_{-2}$, en el qual cada x_i és un dígit binari, el nombre $3_{(10)}$ s'ha de representar com a $011,00_{(2)}$.

Un **format** de representació numèrica és la manera específica com s'han de representar els valors numèrics amb què treballem.

El format fixa el conjunt de nombres que es poden representar.

Terminologia

Al llarg del text, farem servir indistintament **representació** i **codificació** per a referir-nos a la seqüència de dígitos associada a un valor numèric en un sistema de representació numèrica.

Nombres representables

Els nombres que es poden representar de manera exacta reben el nom de *nombres representables*.

En els subapartats següents descrivim els paràmetres que ens ajuden a mesurar l'eficiència d'un format de representació numèrica: el rang de representació, la precisió i l'error de representació.

2.1.1. Rang de representació

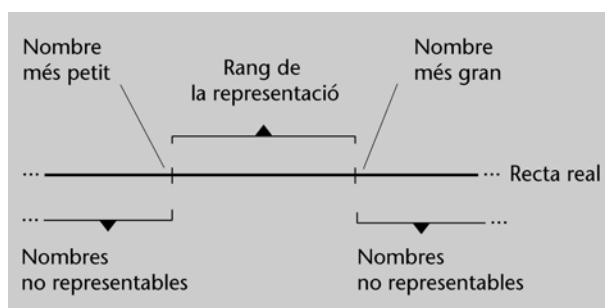
Si fixem el format $x_1x_0,x_{-1}x_{-2}$ a la base 10, només podem representar nombres entre el $00,00_{(10)}$ (el nombre més petit representable en aquest format) i el $99,99_{(10)}$ (el nombre més gran representable en aquest format). El nombre $935_{(10)}$ no es pot representar en aquest format ja que no és dins l'interval de representació. Els nombres que es poden representar en un format estan delimitats dins d'un **interval** que rep el nom de **rang**.

Atenció

En un format només podem representar un conjunt de nombres. En un format amb rang $[0, 99,99]$, el nombre $34,789_{(10)}$ no es pot representar de manera exacta, perquè té 3 dígits fraccionaris.

El **rang** d'un format de representació numèrica és l'interval més petit que conté tots els nombres representables. Els límits de l'interval els determina el nombre més gran i el nombre més petit que s'hi poden representar.

La notació que es fa servir per a definir un rang és $[a,b]$, en la qual a i b són els límits de l'interval en decimal i en formen part.



Els nombres que són fora del rang de representació d'un format no són representables en aquell format. !

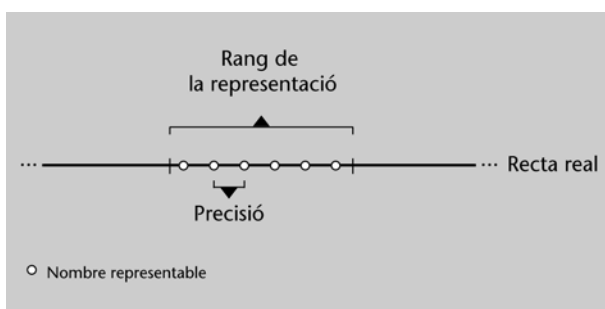
Hi ha una limitació inherent al nombre de bits disponibles en un format de representació: amb n dígits en base b , disposem de b^n codis o combinacions de dígits. Cada una d'aquestes combinacions pot representar un valor numèric. Per tant, amb n dígits en base b podrem representar un màxim de b^n nombres diferents. !

Amb 5 bits disposem de $2^5 = 32$ combinacions diferents. Podrem representar 32 nombres. La codificació que es faci servir determinarà quins són aquests nombres. En base 10 i 4 dígits disposem de 10^4 codis diferents (combinacions dels 4 dígits decimals). Si fem aquests codis per a representar nombres naturals, podrem representar des del 0000 (0) fins al 9999 ($10^4 - 1$).

2.1.2. Precisió

Estem acostumats a treballar de manera dinàmica amb la precisió i l'ajustem automàticament a les nostres necessitats. Per tal de mesurar la longitud d'una taula en metres, per exemple, treballem amb dos dígits fraccionaris. Un format d'aquestes característiques ens permetrà distingir 1,52 m de 1,53 m, però no de 1,5234 m. Diem que la precisió d'aquest format és de 0,01 m, que és la distància entre dos valors consecutius representables en aquest format.

La **precisió** d'un format de representació numèrica és la distància entre dos nombres representables consecutius.



Nota

En la majoria de formats de representació la distància entre dos nombres representables consecutius qualssevol és la mateixa.

2.1.3. Error de representació

En un format de 4 dígits decimals, dels quals 2 són fraccionaris, els nombres són de la forma $x_1x_0,x_{-1}x_{-2}$, en la qual x_i és un dígit decimal. Hi podem representar el $12,34_{(10)}$ o el $45,20_{(10)}$ de manera exacta, però no el $15,026_{(10)}$. Si hem de treballar amb aquest nombre haurem de fer servir una **aproximació**. Podem aproximar-lo per un nombre representable proper com el $15,03_{(10)}$. Treballar amb una aproximació comporta cometre un error. En aquest cas, l'error que cometem és $15,03_{(10)} - 15,026_{(10)} = 0,004_{(10)}$.

L'**error de representació** ε és la distància entre el nombre X que volem representar i el nombre representable \hat{X} al qual l'aproximem. És a dir,

$$\varepsilon = |X - \hat{X}|.$$

Rangs de representació

En el format $x_1x_0,x_{-1}x_{-2}$ en base 10, el rang de representació és $[0, 99,99]$. Un nombre com el $128_{(10)}$, que està fora del rang de representació, no és representable. No es considera que $99,99_{(10)}$ sigui una aproximació vàlida per a aquest nombre en aquest format.

Els nombres que no són dins del rang de representació del format no són representables ni aproximables.

2.1.4. Aproximacions: truncament i arrodoniment

En el format $x_1x_0,x_{-1}x_{-2}$ en base 10, tant el $23,45_{(10)}$ com el $23,46_{(10)}$ són aproximacions vàlides del $23,457_{(10)}$. Hem de triar una de les dues possibilitats, per

la qual cosa establim un criteri de tria. Aquest procés de tria l'anomenem **aproximació** o **quantificació**. Els criteris de tria més habituals són el **truncament** i l'**arrodoniment**.

1) Truncament

El truncament és el criteri de quantificació més directe i senzill d'aplicar, ja que no comporta cap tipus de càlcul i consisteix a ignorar els dígits que sobren. En el format $x_1x_0,x_{-1}x_{-2}$ en base 10, aquest criteri aproxima el nombre $23,457_{(10)}$ pel $23,45_{(10)}$ com a conseqüència del fet d'ignorar el darrer dígit, que no cap en el format.

La quantificació o aproximació per **truncament** consisteix en menystenir els dígits fraccionaris que no caben en el format. El procés de truncament no comporta cap tipus de càlcul.

Truncament

El gran avantatge del truncament és que no comporta cap tipus de càlcul aritmètic.

Per truncament al format $x_1x_0,x_{-1}x_{-2}$ en base 10, tant el $23,451_{(10)}$, el $23,456_{(10)}$ com el $23,459_{(10)}$ s'aproximen pel $23,45_{(10)}$. Ara bé, l'error comès en cada cas és diferent. L'error és 0,001 per al $23,451_{(10)}$ (ja que $|23,451_{(10)} - 23,45_{(10)}| = 0,001$), 0,006 per al $23,456_{(10)}$ i 0,009 per al $23,459_{(10)}$. En tots els casos l'error de representació és inferior a la precisió, que és 0,01.

En una aproximació per truncament, l'**error màxim** de representació és inferior a la precisió del format de representació.

2) Arrodoniment

El $23,459_{(10)}$ al format $x_1x_0,x_{-1}x_{-2}$ en base 10 s'aproxima per truncament pel $23,45_{(10)}$ i l'error és 0,009. Ara bé, si aproximéssim el $23,459_{(10)}$ pel $23,46_{(10)}$, l'error seria 0,001 ($|23,46_{(10)} - 23,459_{(10)}| = 0,001$), és a dir, un error més petit. El $23,46_{(10)}$ és més proper i seria més exacte treballar-hi. Aquesta aproximació rep el nom d'**arrodoniment** o **aproximació al més pròxim**.

La quantificació o aproximació per **arrodoniment** consisteix a escollir el nombre representable més proper al nombre que volem representar. El procés d'arrodoniment comporta operacions aritmètiques.

Nota

El nombre que obtenim per truncament coincideix amb el que obtenim per arrodoniment sempre que el nombre resultant per truncament sigui el nombre representable més proper al nombre que volem representar.

Si apliquem el criteri d'arrodoniment al format $x_1x_0,x_{-1}x_{-2}$ en base 10, el $23,451_{(10)}$ s'aproxima pel $23,45_{(10)}$, mentre que el $23,456_{(10)}$ o el $23,459_{(10)}$ s'aproximen pel $23,46_{(10)}$, que els hi és més proper. L'error és 0,001 per al $23,451_{(10)}$, 0,004 per al $23,456_{(10)}$ i 0,001 per al $23,459_{(10)}$. L'error comès és inferior a la meitat de la precisió, és a dir, inferior a 0,005 en aquest cas.

En una aproximació per arrodoniment, l'**error màxim** de representació és igual a la meitat de la precisió del format de representació.

Una manera senzilla d'aplicar l'arrodoniment al nombre $23,459_{(10)}$ al format $x_1x_0x_{-1}x_{-2}$ en base 10 és sumar-li la meitat de la precisió (és a dir, $0,005$) i, tot seguit, fer el truncament del resultat: $23,459_{(10)} + 0,005_{(10)} = 23,464_{(10)}$, que truncat a dos dígits fraccionaris és el $23,46_{(10)}$.

Per aproximar un nombre per arrodoniment hem de fer el següent:

- 1) Sumar la meitat de la precisió del format de representació al nombre al qual el volem aproximar.
- 2) Truncar el resultat de la suma segons el nombre de dígits fraccionaris disponibles al format de representació.

Aproximació per arrodoniment

Per aproximar per arrodoniment el nombre $1,526264_{(10)}$ al format $x_1x_0x_{-1}x_{-2}x_{-3}x_{-4}$ en base 10, procedirem de la manera següent:


- 1) Sumar la meitat de la precisió del format de representació al nombre que volem aproximar:

$$1,526264_{(10)} + 0,00005_{(10)} = 1,526314_{(10)}$$

- 2) Truncar el resultat de la suma segons el nombre de dígits fraccionaris disponibles al format de representació:

$$1,526314_{(10)} \rightarrow 1,5263_{(10)}$$

Per tant, el nombre $1,526264_{(10)}$ l'aproximem per arrodoniment en aquest format pel $1,5263_{(10)}$.

L'inconvenient de l'arrodoniment és que, a diferència del truncament, comporta operacions aritmètiques. 

2.1.5. Sobreiximent

En fer operacions aritmètiques amb nombres en un format determinat, ens podem trobar que el resultat estigui fora del rang de representació. És el que es coneix com **sobreiximent**.

El **sobreiximent** apareix quan el resultat d'una operació supera el rang de representació.

Terminologia

En anglès, el terme *sobreiximent* rep el nom d'**overflow**.

Dins dels computadors, els nombres naturals es representen en base 2, la precisió és 1 (ja que no hi ha bits fraccionaris) i el rang depèn del nombre de bits disponibles en el format.

El **rang** de representació dels nombres naturals en un format de n bits és, en decimal, $[0, 2^n - 1]$ i la seva **precisió** és 1.


El rang de representació es pot ampliar si augmentem el nombre de bits de la representació. L'ampliació del nombre de bits d'un format de representació rep el nom d'**extensió**.

L'**extensió** dels nombres naturals representats en un format de n bits a un format de m bits, amb $m > n$, l'aconseguim si afegim, a l'esquerra de la codificació, els zeros necessaris fins a completar els m bits del format nou.

La representació del nombre natural $10_{(10)}$ en base 2 en un format de 5 bits és $01010_{(2)}$. L'extensió d'aquesta codificació a un format de 8 bits l'aconseguim si afegim zeros a l'esquerra fins a completar els 8 bits del format nou, amb la qual cosa la nova codificació serà $00001010_{(2)}$.

Les operacions de suma i de resta segueixen les pautes exposades anteriorment. Si es produeix un ròssec a la darrera etapa de suma, hi ha sobreiximent:

$$\begin{array}{r}
 \begin{array}{ccccccc}
 1 & 1 & & & & & \\
 | & & & & & & \\
 1 & 1 & 1 & 0 & 1 & 0 & \\
 | & & & & & & \\
 + & 0 & 1 & 0 & 1 & 0 & 1 \\
 \hline
 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
 | & & & & & & \\
 \uparrow & & & & & &
 \end{array}
 \end{array}
 \begin{array}{l}
 \leftarrow \text{ròssecs} \\
 \\
 \\
 \\
 \leftarrow \text{resultat} \\
 \text{sobreiximent}
 \end{array}$$

La suma de dos nombres naturals de n bits dona lloc a un resultat que com a màxim necessita $n + 1$ bits per a representar-lo. 


El **sobreiximent** en la suma de dos nombres naturals es produeix quan tenim un ròssec a la darrera etapa de suma. L'operació de resta de nombres naturals no pot donar lloc a sobreiximent.

Les operacions de multiplicació i divisió entera per potències de la base de numeració s'ajusten als procediments ja descrits.

Canvi de base del nombre $10_{(10)}$ a base 2


Seguint el mètode basat en el teorema de la divisió entera:

$$\begin{array}{l}
 10 = 5 \cdot 2 + 0 \\
 5 = 2 \cdot 2 + 1 \\
 2 = 1 \cdot 2 + 0 \\
 1 = 0 \cdot 2 + 1 \\
 \hline
 10_{(10)} = 1010_{(2)}
 \end{array}$$

Vegeu la suma i la resta en els sistemes posicionals en el subapartat 1.6 i 1.7. 

Atenció

La resta de dos naturals no pot produir sobreiximent perquè restem la magnitud petita de la gran. Restar la magnitud gran de la petita no és una operació vàlida en els naturals, perquè el resultat seria un nombre amb signe.

Vegeu la multiplicació i divisió per potències de la base en el subapartat 1.8. 

La divisió entera per una potència de la base no produeix sobreeximent, perquè el resultat són dos nombres naturals (quocient i residu) més petits que el dividend. En la multiplicació hi ha sobreeximent si el resultat supera el rang del format. !

La divisió de dos naturals

L'operació de divisió en nombres naturals ha de ser la divisió entera atès que els nombres naturals no tenen part fraccionària.

2.3. Nombres enters

Els enters són els nombres amb signe i sense part fraccionària, incloent-hi el zero: $\dots -3, -2, -1, 0, +1, +2, +3\dots$ Es diferencien dels naturals per la presència d'un signe que indica si la magnitud és positiva o negativa. Aquest signe es pot incorporar a la codificació dels nombres dins dels computadors de diverses maneres. En els apartats següents descrivim les més utilitzades en els computadors: signe i magnitud, i complement a 2.

2.3.1. Representació d'enters en signe i magnitud en base 2

Signe i magnitud és, probablement, la manera més intuïtiva de representar nombres amb signe. En **signe i magnitud**, el bit més significatiu (MSB) emmagatzema el signe i la resta codifica la magnitud. Un 1 en el dígit més significatiu indica signe negatiu, mentre que un 0 indica signe positiu.

Així, si la cadena de bits 101001 és un nombre en signe i magnitud, sabrem que és un nombre negatiu, perquè el bit més significatiu és 1, i que la magnitud és $01001_{(2)}$, que en base 10 és el $9_{(10)}$. Aquesta cadena de bits codifica el $-9_{(10)}$.

MSB i LSB

MSB és l'abreviació de *most significant bit*, és a dir, el bit més significatiu de la representació, que es correspon amb el dígit de l'extrem esquerre. *LSB* és l'abreviació de *least significant bit*, és a dir, el bit menys significatiu de la representació, que es correspon amb el dígit de l'extrem dret.

Un nombre codificat en **signe i magnitud** amb n bits el determina la cadena de bits $x_{n-1}x_{n-2}\dots x_1x_0$, en la qual x_{n-1} codifica el signe i $x_{n-2}\dots x_1x_0$ la magnitud. El signe és positiu si x_{n-1} és 0, i negatiu si és 1.

Al llarg del text farem servir la notació $X_{(SM2)}$ per a identificar un nombre codificat en signe i magnitud en base 2. !

La codificació en signe i magnitud també es fa servir per a nombres fraccionaris amb signe, tal com expliquem més endavant. !

Representació en signe i magnitud

Per representar el $-12_{(10)}$ en signe i magnitud, 6 dígits i base 2, hem de passar la magnitud $12_{(10)}$ a base 2 ($12_{(10)} = 1100_{(2)}$) i posar el bit més significatiu de la representació (el bit de més a l'esquerra) a 1. La representació amb 6 bits és, doncs, $101100_{(SM2)}$.

El $+12_{(10)}$ es representa en el mateix format com $001100_{(SM2)}$. Només canvia el bit més significatiu, perquè la magnitud és la mateixa.

Canvi de base del nombre $12_{(10)}$ a base 2

Si apliquem la divisió entera:

$$\begin{array}{rcl} 12 & = & 6 \cdot 2 + 0 \\ 6 & = & 3 \cdot 2 + 0 \\ 3 & = & 1 \cdot 2 + 1 \\ 1 & = & 0 \cdot 2 + 1 \end{array} \quad \uparrow$$

$12_{(10)} = 1100_{(2)}$

Rang de representació en signe i magnitud i base 2

El format de signe i magnitud és simètric, és a dir, podem representar-hi tants valors positius com negatius. Amb 4 bits i signe i magnitud tindrem 1 bit (el més significatiu) per al signe i 3 per a la magnitud:

- Valors possibles per al signe:

0 → +
1 → -

- Valors possibles per a la magnitud:

000 ₍₂₎ = 0 ₍₁₀₎	100 ₍₂₎ = 4 ₍₁₀₎
001 ₍₂₎ = 1 ₍₁₀₎	101 ₍₂₎ = 5 ₍₁₀₎
010 ₍₂₎ = 2 ₍₁₀₎	110 ₍₂₎ = 6 ₍₁₀₎
011 ₍₂₎ = 3 ₍₁₀₎	111 ₍₂₎ = 7 ₍₁₀₎

Avantatges del format de signe i magnitud

El format de signe i magnitud té avantatges a l'hora de fer multiplicacions: s'operen per separat les magnituds i els signes i, posteriorment, s'ajunten els resultats obtinguts de manera independent.

Si combinem el signe i la magnitud, podem representar els valors enters entre -7 i +7. Per tant, el rang de representació és [-7, +7].


En general, en signe i magnitud en base 2, el **rang** d'enters representable amb n bits és, en decimal,

$$[-(2^{n-1} - 1), 2^{n-1} - 1]$$

Si apliquem aquesta expressió al cas de 4 bits tenim:

$$[-(2^{4-1} - 1), 2^{4-1} - 1] = [-(2^3 - 1), 2^3 - 1] = [-7, +7]$$

que és el rang al qual havíem arribat de manera experimental.

La precisió en la codificació d'enters en signe i magnitud és 1, perquè podem codificar tots els enters del rang de representació. 

Si fos necessari ampliar el rang de representació, hauríem de fer una extensió del format de signe i magnitud.

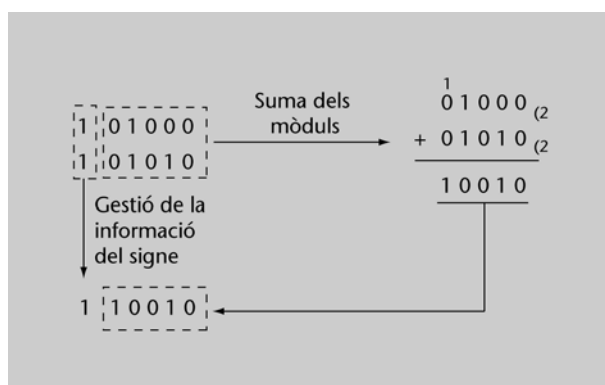
L'**extensió** d' n a m bits dels nombres en signe i magnitud, amb $m > n$, l'aconsegim si afegim, a l'esquerra de la magnitud, els zeros necessaris per a completar els m bits, mantenint el bit de l'extrem esquerre per a la codificació del signe.

Consegüentment, l'enter negatiu $11010_{(SM2)}$ codificat en signe i magnitud i 5 bits es pot estendre a un format de 8 bits si afegim zeros a la dreta del signe, de manera que la codificació d'aquest mateix nombre en el nou format seria $10001010_{(SM2)}$. L'extensió de nombres positius es fa de la mateixa manera. L'extensió a 8 bits de la codificació en signe i magnitud de l'enter positiu $01010_{(SM2)}$ ens porta al $00001010_{(SM2)}$.

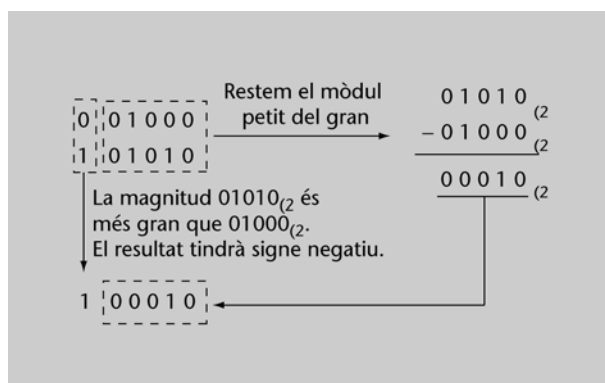
2.3.2. Suma i resta en signe i magnitud

La suma de dos nombres positius o negatius en signe i magnitud és senzilla. Hem de fer la suma de les magnituds i donar al resultat el signe dels operands. La suma de les magnituds pot produir sobreiximent.

La suma dels nombres $101000_{(SM2)}$ i $101010_{(SM2)}$ codificats en signe i magnitud i 6 bits és la següent:



La suma d'un positiu i un negatiu és més complexa: cal analitzar les magnituds per a saber quina és la més gran, restar la magnitud petita de la gran i aplicar al resultat el signe de la magnitud més gran. El procediment de suma dels nombres $001000_{(SM2)}$ i $101010_{(SM2)}$ codificats en signe i magnitud i 6 bits és el següent:



La suma de dos nombres del mateix signe i la resta de nombres del signe contrari pot produir sobreiximent.

En signe i magnitud, hi ha **sobreeiximent** en la suma de dos nombres del mateix signe o en la resta de nombres de signe contrari quan apareix un ròssec a la darrera etapa de suma o resta de les magnituds.

Ni la suma d'un positiu i un negatiu, ni la resta de nombres del mateix signe poden produir sobreeiximent.

En la suma de $101010_{(SM2)}$ i $111010_{(SM2)}$, en signe i magnitud i 6 bits, examinem, en primer lloc, els signes. Són dos nombres negatius, ja que el bit de més pes d'ambdós és 1. Per tant, procedirem a la suma de les magnituds:

$$\begin{array}{r}
 \begin{array}{cccccc}
 1 & 1 & & 1 & & \\
 | & & & & & \\
 & 0 & 1 & 0 & 1 & 0 \\
 + & 1 & 1 & 0 & 1 & 0 \\
 \hline
 1 & 0 & 0 & 1 & 0 & 0
 \end{array}
 \begin{array}{l}
 \leftarrow \text{ròssecs} \\
 \\
 \\
 \leftarrow \text{resultat}
 \end{array}
 \end{array}$$

↑
sobreeiximent

La suma de les magnituds produeix sobreeiximent, ja que tenim un ròssec a la darrera etapa. Per tant, el resultat no cap en el format definit i no el podem representar.

Els inconvenients principals del sistema de signe i magnitud són la complexitat de les operacions de suma i resta i l'existència de dues representacions per al 0: un "0 positiu", quan la magnitud i el signe són 0 i un "0 negatiu" quan la magnitud és 0 i el signe 1.

2.3.3. Representació en complement a 2


El **complement a 2**, abreujat habitualment per **Ca2** o **C2**, és un sistema de representació de nombres amb signe en base 2. Actualment, el Ca2 és el sistema més emprat per a codificar nombres enters en els computadors, perquè presenta dos avantatges: una codificació única pel zero i simplicitat en les operacions de suma i resta.

Els **nombres positius** en Ca2 es codifiquen de la mateixa manera que en signe i magnitud: el bit de l'extrem esquerre és 0, per a indicar signe positiu, i la resta conté la magnitud.

La codificació d'un **nombre negatiu** $-X$ en Ca2 és el resultat en binari de l'operació $2^n - |X|$, en la qual $|X|$ és el valor absolut de X .

Inconvenients del format de Ca2

Sense que afecti a l'eficiència dins dels computadors, el valor de les magnituds negatives codificades en Ca2 són més difícils de reconèixer per a nosaltres.

Al llarg del text farem servir la notació $X_{(Ca2)}$ per a identificar un nombre codificat en complement a 2. 

Codificació de nombres negatius en Ca2

Per trobar la codificació en Ca2 i 6 bits del valor $-11010_{(2)}$, fem l'operació següent:

$$2^6 - |X| = 1000000_{(2)} - 11010_{(2)} = 100110_{(Ca2)}$$

1	0	0	0	0	0	0	0	(2)		
1	1	1	1	1						
-					1	1	0	1	0	(2)
0	1	0	0	1	1	0				(Ca2)

Així, doncs, la codificació en Ca2 i 6 bits del valor $-11010_{(2)}$ és $100110_{(Ca2)}$


La codificació del valor $+11010_{(2)}$ en Ca2 coincideix amb la codificació en signe i magnitud. Tindrem un 0 per al signe i a continuació 5 bits amb la magnitud: el valor $+11010_{(2)}$ es codifica en Ca2 com a $011010_{(Ca2)}$.

Per trobar la codificació en Ca2 i 8 bits del valor $-11010_{(2)}$, fem l'operació següent:

$$2^8 - |X| = 100000000_{(2)} - 11010_{(2)} = 11100110_{(Ca2)}$$

1	0	0	0	0	0	0	0	0	0	(2)	
1	1	1	1	1	1	1					
-						1	1	0	1	0	(2)
0	1	1	1	0	0	1	1	0		(Ca2)	

Així, doncs, la codificació en Ca2 i 8 bits del valor $-11010_{(2)}$ és $11100110_{(Ca2)}$

També podem aconseguir la codificació en Ca2 d'una magnitud negativa mitjançant un canvi de signe a la codificació de la magnitud positiva. Vegeu, més endavant, el subapartat 2.3.4. 

Rang de representació en Ca2

En la taula següent mostrem els enters representables amb 4 bits en signe i magnitud i en complement a 2 i la seva correspondència:

Decimal	Signe i magnitud	Ca2
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
0	0000 1000	0000
-1	1001	1111
-2	1010	1110
-3	1011	1101

Taula

En la taula veiem que els positius es codifiquen igual en Ca2 i signe i magnitud. El rang dels positius és el mateix en els dos sistemes. En canvi, en Ca2 tenim un negatiu més que en signe i magnitud. Això és perquè en Ca2 hi ha una representació única del zero, mentre que en signe i magnitud n'hi ha dues.

Decimal	Signe i magnitud	Ca2
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	no és representable	1000


En Ca2 i 4 bits podem representar des del $-8_{(10)}$ fins al $+7_{(10)}$.

En general, el **rang** d'enters representables amb n bits en Ca2 és en decimal:

$$[-2^{n-1}, 2^{n-1} - 1]$$

Amb 4 bits, el rang és: $[-2^{4-1}, 2^{4-1} - 1] = [-2^3, 2^3 - 1] = [-8, +7]$

En Ca2, per augmentar el nombre de bits en què codifiquem un enter positiu podem seguir el mateix procediment que en signe i magnitud. En canvi, l'extensió per als enters negatius és diferent. El $-10_{(10)}$ en Ca2 i 5 bits és el $10110_{(Ca2)}$, mentre que amb 8 bits es codifica com $11110110_{(Ca2)}$. La diferència entre les codificacions és que en la segona s'han afegit tres 1 a l'esquerra.

Fixem-nos que en tots dos casos els bits que s'afegeixen coincideixen amb el valor del bit de més pes: zeros per als positius i uns per als negatius. 

Exemple

En Ca2, el $-10_{(10)}$ es codifica amb 5 bits per al 10110:
 $2^5 - |-10_{(10)}| = 32_{(10)} - 10_{(10)} = 22_{(10)} = 10110_{(Ca2)}$
 En Ca2, el $-10_{(10)}$ es codifica amb 8 bits per al 11110110:
 $2^8 - |-10_{(10)}| = 256_{(10)} - 10_{(10)} = 246_{(10)} = 11110110_{(Ca2)}$

En Ca2, **per estendre** un format d' n bits a m bits, amb $m > n$, copiem a l'esquerra el bit de més pes les vegades necessàries per completar els m bits. Aquest procés rep el nom d'**extensió del signe**.

En Ca2, el bit de més pes indica el signe

En Ca2, un 1 al bit de més pes indica que el nombre és negatiu, mentre que un 0 indica que és positiu.

2.3.4. Canvi de signe en complement 2

Farem un canvi de signe d'un nombre en Ca2 si seguim els passos següents:

- 1) Fer el complement bit a bit de la codificació en Ca2.
- 2) Sumar 1 al bit menys significatiu de la codificació.

En base 2, el complementari del 0 és l'1 i el de l'1 és el 0. 

Complement bit a bit

Entenem per *complement bit a bit* la substitució de cada bit pel seu complementari.

Canvi de signe en complement a 2

Per fer el canvi de signe del valor numèric $11000110_{(Ca2)}$ (el qual, si seguim el procediment explicat a l'apartat següent, veuríem que es tracta del $-58_{(10)}$, que està codificat en complement a 2 i 8 dígits, fem l'operació següent:

$$\begin{array}{rcl}
 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & (Ca2) & \leftarrow \text{valor numèric inicial} \\
 & & & & & 1 & & & & \leftarrow \text{ròssecs} \\
 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & (Ca2) & \leftarrow \text{complement bit a bit de l'expressió inicial} \\
 + & & & & & & & 1 & & \leftarrow \text{sumem 1 al bit menys significatiu del format} \\
 \hline
 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & (Ca2) &
 \end{array}$$

D'aquesta operació resulta el $00111010_{(Ca2)}$, que codifica la mateixa magnitud, però amb signe positiu.

El canvi de signe d'un nombre en Ca2 també el podem aconseguir si examinem els bits de dreta a esquerra i:

- 1) Mantenim els mateixos bits fins que trobem el primer 1 (aquest inclòs).
- 2) Fem el complement bit a bit de la resta.

Canvi de signe en complement a 2

Per fer el canvi de signe del 11000110 que està Ca2 i 8 bits, l'examinem de dreta a esquerra, fent el complement bit a bit després del primer 1:

```

11000110
  ^ es mantenen els bits fins aquí (inclòs el primer 1 que hi trobem)
  10
  ^ es complementen els bits a partir d'aquest punt
001110

```

D'aquesta operació en resulta el 00111010 , que codifica la mateixa magnitud però amb signe positiu.

El canvi de signe del 00011110 que està en Ca2 i 8 bits l'obtenim seguint el mateix procediment:

```

00011110
  ^ es mantenen els bits fins aquí (primer 1 que trobem, inclòs)
  10
  ^ es complementen els bits a partir d'aquest punt
111000

```

El resultat és 11100010 , que codifica la mateixa magnitud però amb signe negatiu.

El canvi de signe es pot fer servir com a alternativa a l'operació $2^n - |X|$ per tal de trobar la codificació en Ca2 d'una magnitud negativa.

La codificació en Ca2 d'una magnitud negativa la podem obtenir si apliquem un canvi de signe a la codificació en Ca2 de la magnitud positiva.

L'operació és reversible. Si apliquem un canvi de signe a la codificació en Ca2 d'una magnitud negativa, trobarem la codificació de la positiva.

2.3.5. Magnitud dels nombres en complement a 2

Com en signe i magnitud, podem saber la magnitud decimal d'un nombre positiu codificat en Ca2 si apliquem el TFN:

$$0101_{(2)} = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = +5_{(10)}$$

En Ca2, la cadena de bits 0101 codifica el valor $+5_{(10)}$.

Per a trobar la magnitud decimal d'un nombre negatiu en Ca2, disposem de dues alternatives:

1) Apliquem el TFN, com en el cas del positiu, però considerant que el bit de més pes és negatiu.

Magnitud decimal d'un valor negatiu codificat en Ca2 amb aplicació del TFN

Apliquem el TFN per trobar la magnitud decimal que codifica la cadena de bits 10001010 en Ca2, considerant que el primer bit és negatiu:

$$\begin{aligned} 10001010_{(Ca2)} &= -1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = \\ &= -128 + 10 = -118_{(10)} \end{aligned}$$

El 10001010 en Ca2 codifica el valor decimal -118 .

2) Apliquem un canvi de signe a la representació en Ca2 del valor negatiu i trobem la magnitud positiva.

Magnitud decimal d'un valor negatiu en Ca2 per canvi de signe

Per conèixer la magnitud decimal que codifica el 10001010 en Ca2:

1) Apliquem un canvi de signe:

1 0 0 0 1 0 1 0	(Ca2)	← Valor numèric inicial
1		← Ròssecs
0 1 1 1 0 1 0 1	(Ca2)	← Complement bit a bit de l'expressió inicial
+	1	← Sumem 1 al bit menys significatiu del format
0 1 1 1 0 1 1 0	(Ca2)	

2) Apliquem el TFN al resultat:

$$01110110_{(Ca2)} = 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = +118_{(10)}$$

Per tant, la cadena de bits 10001010 codifica en Ca2 l'enter decimal -118 .

2.3.6. Suma en complement a 2

El mecanisme de suma en Ca2 és el mateix que el que es fa servir en qualsevol altre sistema posicional. Hem de saber reconèixer, però, quan es produeix sobreiximent.

! Vegeu la suma i la resta en els sistemes posicionals en els subapartats 1.6 i 1.7.

Suma de dos valors positius en Ca2

Considerem la suma següent de dos nombres positius en Ca2 i 6 bits:

$$\begin{array}{rcccccccl}
 0 & 0 & 1 & 0 & 1 & 0 & (\text{Ca2}) & \rightarrow & & +10 & (10) \\
 + & 0 & 0 & 0 & 1 & 0 & 1 & (\text{Ca2}) & \rightarrow & + & +5 & (10) \\
 \hline
 0 & 0 & 1 & 1 & 1 & 1 & (\text{Ca2}) & \rightarrow & & +15 & (10) \\
 & & & & & & & & & & & (\text{resultat correcte})
 \end{array}$$

Podem trobar la correspondència entre els nombres en Ca2 i els valors decimals si apliquem qualsevol dels dos mètodes que s'exposen en l'apartat 2.3.5.

Sabem que el resultat és correcte perquè hem fet la suma de dos positius i n'obtenim una magnitud positiva. Quan el resultat supera el rang de representació, la suma de dos positius genera una magnitud negativa, com en el cas següent:

$$\begin{array}{rcccccccl}
 1 & 1 & 1 & 1 & & & \leftarrow \text{ròssecs} & & & & \\
 0 & 1 & 0 & 1 & 1 & 0 & (\text{Ca2}) & \rightarrow & & +22 & (10) \\
 + & 0 & 0 & 1 & 1 & 1 & 1 & (\text{Ca2}) & \rightarrow & + & +15 & (10) \\
 \hline
 1 & 0 & 0 & 1 & 0 & 1 & (\text{Ca2}) & \rightarrow & & -27 & (10) \\
 & & & & & & & & & & & (\text{sobreeiximent})
 \end{array}$$

En Ca2 i 6 bits, el rang és $[-2^{6-1}, +2^{6-1} - 1] = [-32, +31]$. El resultat d'aquesta suma hauria de ser $+37_{(10)}$ ($22_{(10)} + 15_{(10)} = +37_{(10)}$), que queda fora del rang.

Hi ha **sobreeiximent** en la suma de dos nombres positius codificats en Ca2 quan el resultat és negatiu.

Suma de dos valors negatius en Ca2

De manera anàloga, el resultat de la suma de dos negatius en Ca2 és correcte quan n'obtenim una magnitud negativa i erroni quan n'obtenim una de positiva.

Considerem la suma de dos nombres negatius en Ca2 i 6 bits següent:

$$\begin{array}{rcccccccl}
 1 & | & 1 & & & & \leftarrow \text{ròssecs} & & & & \\
 & | & 1 & 1 & 1 & 0 & 1 & 0 & (\text{Ca2}) & \rightarrow & & -6 & (10) \\
 + & | & 1 & 1 & 0 & 1 & 0 & 1 & (\text{Ca2}) & \rightarrow & + & -11 & (10) \\
 \hline
 1 & | & 1 & 0 & 1 & 1 & 1 & 1 & (\text{Ca2}) & \rightarrow & & -17 & (10) \\
 & | & & & & & & & & & & & (\text{resultat correcte})
 \end{array}$$

Podem trobar la correspondència entre els nombres en Ca2 i els valors decimals si apliquem qualsevol dels dos mètodes que exposen en l'apartat 2.3.5.

El resultat és correcte, encara que hi hagi un ròssec en la darrera etapa de la suma, perquè obtenim una magnitud negativa en la suma de dos negatius. En canvi, el resultat de la suma següent és erroni:

1		1	1	1		← ròssecs			
		1	0	0	1	1	0	(Ca2)	→
		1	0	1	1	1	1	(Ca2)	→
+		1	0	1	1	1	1	(Ca2)	→
		0	1	0	1	0	1	(Ca2)	→
		0	1	0	1	0	1	(Ca2)	→
		-26							(10)
		+ -17							(10)
		+21							(10)
		(sobreeiximent)							

Com en el cas anterior, el ròssec de la darrera etapa s'ha de menystenir, però sumant dos negatius n'hem obtingut un de positiu. Per tant, hi ha sobreeiximent.

Hi ha **sobreiximent** en la suma de dos nombres negatius codificats en Ca2 quan el resultat és positiu. El ròssec de la darrera etapa no indica sobreiximent i s'ha de menystenir en tots els casos.

Suma d'un valor positiu i un valor negatiu en Ca2

El resultat de la suma d'un positiu i un negatiu en Ca2 sempre és correcte.

$$\begin{array}{r}
 1 \mid 1 \qquad \qquad \qquad \leftarrow \text{r\`ossecs} \\
 \mid 1 \ 1 \ 1 \ 0 \ 1 \ 0 \quad (\text{Ca2} \rightarrow -6 \quad (10) \qquad 1 \ 0 \ 0 \ 1 \ 1 \ 0 \quad (\text{Ca2} \rightarrow -26 \quad (10) \\
 + \mid 0 \ 1 \ 0 \ 1 \ 0 \ 1 \quad (\text{Ca2} \rightarrow +21 \quad (10) \quad + \mid 0 \ 1 \ 0 \ 0 \ 0 \ 1 \quad (\text{Ca2} \rightarrow +17 \quad (10) \\
 \hline
 1 \mid 0 \ 0 \ 1 \ 1 \ 1 \ 1 \quad (\text{Ca2} \rightarrow +15 \quad (10) \quad \hline
 1 \ 1 \ 0 \ 1 \ 1 \ 1 \quad (\text{Ca2} \rightarrow -9 \quad (10) \\
 \mid \qquad \qquad \qquad \text{(resultat correcte)} \qquad \qquad \qquad \text{(resultat correcte)}
 \end{array}$$

Podem trobar la correspondència entre els nombres en Ca_2 i els valors decimals si apliquem qualsevol dels dos mètodes que exposem en l'apartat 2.3.5.

L'aparició de ròssec en la darrera etapa, com a l'operació de l'esquerra, no indica sobreeximent i s'ha de menystenir.

La suma d'un nombre positiu i un nombre negatiu codificats en Ca2 no pot produir sobreeximent. El ròssec que es pot produir a la darrera etapa no indica sobreeximent i s'ha de menystenir en tots els casos.

2.3.7. Resta en complement a 2

El procediment que seguim per fer la resta en Ca2 és aplicar un canvi de signe en el subtrahend (operand que volem restar) i fer una operació de suma. Sumem el minuend amb el subtrahend en el qual hem canviat el signe.

L'operació de resta en Ca2 es redueix a una operació de suma un cop hem canviat el signe del subtrahend.

La resta $011010_{(Ca2)} - 001011_{(Ca2)}$ ($26_{(10)} - 11_{(10)}$) ens servirà d'exemple per a il·lustrar el procediment:

1) Apliquem un canvi de signe al subtrahend:

a) Fem el complement bit a bit de 001011, amb el qual obtenim 110100.

b) Sumem 1 al bit menys significatiu:

$$\begin{array}{r} 1\ 1\ 0\ 1\ 0\ 0\ (Ca2) \\ + \qquad\qquad\qquad 1\ (Ca2) \\ \hline 1\ 1\ 0\ 1\ 0\ 1\ (Ca2) \end{array}$$

2) Fem l'operació de suma:

$$\begin{array}{r} \begin{array}{c} | \\ 1\ |1 \\ |0\ 1\ 1\ 0\ 1\ 0\ (Ca2) \\ +\ |1\ 1\ 0\ 1\ 0\ 1\ (Ca2) \\ \hline 1\ |0\ 0\ 1\ 1\ 1\ 1\ (Ca2) \\ | \end{array} \quad \begin{array}{l} \leftarrow \text{ròssecs} \\ \rightarrow \\ \rightarrow \\ \rightarrow \end{array} \quad \begin{array}{l} \\ +26_{(10)} \\ +\ -11_{(10)} \\ +15_{(10)} \end{array} \\ \text{(resultat correcte)} \end{array}$$

Podem trobar la correspondència entre els nombres en Ca2 i els valors decimals si apliquem qualsevol dels dos mètodes que exposem en l'apartat 2.3.5.

El resultat és correcte. El ròssec de la darrera etapa s'ha de menystenir i no es produeix sobreiximent. Recordem que la suma d'un nombre positiu i un nombre negatiu no pot donar lloc a sobreiximent.

2.3.8. Multiplicació per 2^k de nombres en complement a 2

Com hem vist, multiplicar per 2^k en sistemes de numeració posicionals de base 2 equival a desplaçar la coma fraccionària k posicions a la dreta. En el cas dels enters, aquest efecte l'aconsegüim afegint k zeros a la dreta:

$$000101_{(Ca2)} \cdot 2^2 = 010100_{(Ca2)} \quad (\text{en decimal, } +5_{(10)} \cdot 2^2 = +20_{(10)})$$

El procediment també és vàlid per a nombres negatius en Ca2:

$$111011_{(Ca2)} \cdot 2^2 = 101100_{(Ca2)} \quad (\text{en decimal, } -5_{(10)} \cdot 2^2 = -20_{(10)})$$

El resultat de **multiplicar per 2^k un nombre en Ca2** l'aconsegüim si afegim k zeros a la dreta.

Vegeu la multiplicació i la divisió per potències de la base de numeració en el subapartat 1.8 d'aquest mòdul.

Per aplicació del TFN

$$\begin{aligned} 000101_2 &= 2^2 + 2^0 = 5_{(10)} \\ 010100_2 &= 2^4 + 2^2 = 20_{(10)} \\ 111011_2 &= -2^5 + 2^4 + 2^3 + 2^1 + 2^0 = -5_{(10)} \\ 101100_2 &= -2^5 + 2^3 + 2^2 = -20_{(10)} \end{aligned}$$

Un cop hem fixat un format d' n bits, afegir k zeros a la dreta ens obliga a perdre k bits de l'esquerra, la qual cosa pot produir sobreiximent. En Ca2 i 6 bits, les operacions següents produeixen sobreiximent:

$$\begin{aligned} 000101_{\text{Ca2}} \cdot 2^4 &= 010000_{\text{Ca2}} & (\text{en decimal, } +5_{(10)} \cdot 2^4 &= +16_{(10)}) \\ 111000_{\text{Ca2}} \cdot 2^4 &= 000000_{\text{Ca2}} & (\text{en decimal, } -8_{(10)} \cdot 2^4 &= 0_{(10)}) \end{aligned}$$

Es produeix **sobreiximent** en multiplicar un nombre en Ca2 per 2^k quan canvia el bit de signe o si es perd un o més bits significatius. Els bits significatius són, per als positius, els 1 i, per als negatius, els 0.

Activitats

21. Convertiu els valors decimals següents a binaris en els sistemes de representació de signe i magnitud i complement a 2, amb un format enter de 8 bits:

- a) 53
- b) -25
- c) 93
- d) -1
- e) -127
- f) -64

22. Si tenim els nombres binaris 00110110, 11011010, 01110110, 11111111 i 11100100, quins són els equivalents decimals si considerem que són valors binaris representats en signe i magnitud?

23. Repetiu l'exercici anterior considerant que les cadenes de bits són nombres en complement a 2.


24. Si tenim les cadenes de bits següents $A = 1100100111$, $B = 1000011101$ i $C = 0101011011$, feu les operacions que proposem a continuació considerant que són nombres binaris en format de signe i magnitud: $A + B$, $A - B$, $A + C$, $A - C$, $B - C$, $B + C$.

25. Repetiu l'activitat anterior considerant que les cadenes representen nombres en complement a 2.

26. Si tenim la cadena de bits 10110101, feu les conversions següents:

- a) Considerant que representa un nombre en Ca2, representeu el mateix nombre en signe i magnitud i 16 bits.
- b) Considerant que representa un nombre en signe i magnitud, representeu el mateix nombre en Ca2 i 16 bits.

2.4. Nombres fraccionaris

Els nombres fraccionaris són els que tenen una part més petita que la unitat, com ara el $0,03_{(10)}$ o el $15,27_{(10)}$. La representació dels nombres fraccionaris dins dels computadors se sol fer destinant un nombre fix de bits, del total de bits del format, a la representació de la part fraccionària. Aquest tipus de representació rep el nom genèric de **representació de coma fixa**. 

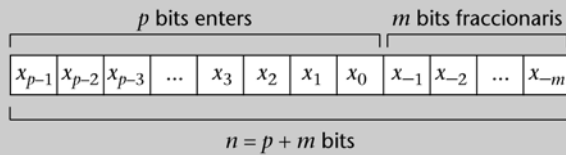
Nombre decimal

Fem servir l'expressió *nombre decimal* per a designar un nombre en base 10, no un nombre amb part fraccionària.

Representació binària en coma fixa

Les representacions de coma fixa no emmagatzemen la posició de la coma de manera explícita. És en la definició del format on s'especifica la posició de la coma, i s'assumeix que sempre és la mateixa.

En un format de representació de coma fixa de n bits ($n = p + m$), en què m bits són fraccionaris, els nombres representats són de la forma:

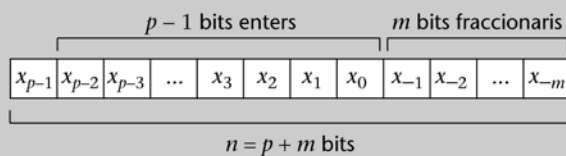


La magnitud decimal del nombre fraccionari representat és $X_{(10)} = \sum_{i=p-1}^{-m} x_i \cdot 2^i$, en què x_i és el bit de la posició i -èsima.

Signe i magnitud en coma fixa

Les magnituds fraccionàries també poden portar associat un signe. En coma fixa, el més habitual és treballar amb una representació de signe i magnitud.

En un format de coma fixa de n bits ($n = p + m$), en el qual m bits són fraccionaris, i en signe i magnitud, els nombres són de la forma:

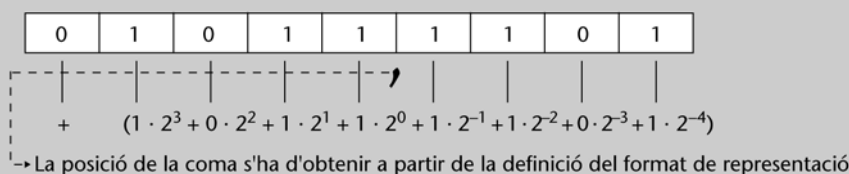


en què x_{p-1} és el bit de signe i la resta de bits codifiquen la magnitud.

Altres maneres de representar nombres

Hi ha altres maneres de representar nombres fraccionaris amb signe. Per exemple, es podria emprar la representació de complement a la base (Ca2 en binari), però no és habitual.

Per conèixer en decimal, el valor numèric representat per la codificació 01011101 que està en signe i magnitud en un format de 9 bits ($n = 9$), dels quals 4 són fraccionaris ($m = 4$), hem d'aplicar el TFN:



Format en coma fixa

Aquesta manera de representar els nombres fraccionaris és una extensió directa de la representació en signe i magnitud de nombres enters. Per tant, presenta els mateixos avantatges i inconvenients. Així, per exemple, el zero té dues representacions, una amb signe negatiu i una altra amb signe positiu.

Per tant, el nombre codificat és $+1011,1101_2 = 11,8125_{(10)}$.

Magnitud decimal d'un nombre codificat en coma fixa i signe i magnitud

Per trobar la magnitud decimal que representa la codificació 10010010 que està en un format de 8 bits, 4 dels quals són fraccionaris i en signe i magnitud, farem les operacions següents:

1) Separem el bit de signe que és 1 (bit de l'extrem esquerre) i que indica signe negatiu. La resta de bits 0010010 codifiquen la magnitud.

2) El valor decimal de la magnitud el podem saber si apliquem el TFN:

$$0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} = 1 + 0,125 = 1,125_{(10)}$$

Per tant, el nombre representat en decimal és $-1,125_{(10)}$.

Codificació d'un valor decimal en coma fixa i signe i magnitud

Per codificar el nombre $-14,75_{(10)}$ en un format de coma fixa de 8 bits en el qual 3 són fraccionaris i signe i magnitud, farem les operacions següents:

1) Canviar a base 2 el nombre $14,75_{(10)}$, seguint el mètode basat en el teorema de la divisió entera.

a) Codificar en binari la part entera ($14_{(10)}$) en 4 bits (ja que dels 8 bits, 3 són fraccionaris i 1 codifica el signe; en resten 4 per a la part entera) aplicant l'algorisme de divisions successives:

$$\begin{array}{l} 14 = 7 \cdot 2 + 0 \\ 7 = 3 \cdot 2 + 1 \\ 3 = 1 \cdot 2 + 1 \\ 1 = 0 \cdot 2 + 1 \end{array} \quad \uparrow$$

Amb la qual cosa obtenim que $14_{(10)} = 1110_{(2)}$.

b) Codificar en binari la part fraccionària en 3 bits:

$$\begin{array}{l} 0,75 \cdot 2 = 1,50 = 1 + 0,5 \\ 0,50 \cdot 2 = 1,0 = 1 + 0,0 \end{array} \quad \downarrow$$

Amb la qual cosa obtenim que $0,75_{(10)} = 0,110_{(2)}$


c) Ajuntar les parts entera i fraccionària en el format de 7 bits 3 dels quals són fraccionaris:

$$14,75_{(10)} = 1110,110_{(2)}$$


2) Afegir el bit de signe a la magnitud. El bit de signe és 1, ja que el signe és negatiu. La representació en un format de coma fixa de 8 bits en el qual 3 són fraccionaris i signe i magnitud del $-14,75_{(10)}$ és el següent:


$$-14,75_{(10)} = 11110,110_{(SM2)}$$

Recordem que la coma no s'emmagatzema sinó que, un cop especificat el format, se'n coneix la posició. En realitat, un computador emmagatzemaria el codi 11110110 sense coma ni especificació de base, que estan fixats a la definició del format.

Quan la part fraccionària excedeix el nombre de bits fraccionaris disponibles al format en signe i magnitud definit, el nombre no es pot representar de manera exacta. Caldrà aplicar un dels mètodes d'aproximació explicats: el truncament o l'arrodoniment. 

A tall d'exemple, intentem representar el nombre $+8,9453125_{(10)}$ en un format de coma fixa i signe i magnitud, 8 bits dels quals 3 són fraccionaris. Si fem el canvi de base, trobem que $8,9453125_{(10)} = 1000,1111001_{(2)}$. Haurem

 Vegeu el mètode basat en el teorema de la divisió entera en el subapartat 1.3.2 d'aquest mòdul.

 Vegeu les aproximacions per truncament i arrodoniment en el subapartat 2.1.4 d'aquest mòdul.

de fer servir un dels mètodes d'aproximació, ja que la part fraccionària no cap en els 3 bits disponibles en el format:

- Per **truncament**: es tracta, senzillament, de menystenir els bits que no hi caben. El $1000,1111001_2$ s'aproximarà pel $1000,111_2$. Afegim el bit de signe i la codificació final serà 01000111 . L'error de representació que es comet en aquest cas és:

$$|1000,1111001_2 - 1000,111_2| = 0,0001001_2 = 0,0703125_{(10)}$$

- Per **arrodoniment**: sumem la meitat de la precisió:

$$1000,1111001_2 + 0,0001_2 = 1001,0000001_2$$

Tot seguit trunquem a 3 bits fraccionaris, de manera que el $1000,1111001_2$ s'aproximarà per al $1001,000_2$. Afegim el bit de signe i la codificació final serà 01001000 . En aquest cas, l'error de representació que es comet és més petit:

$$|1000,1111001_2 - 1001,000_2| = 0,0000111_2 = 0,0546875_{(10)}$$

Rang i precisió en coma fixa

La magnitud binària més gran que podem representar en un format de coma fixa és la que obtenim si posem tots els bits que representen la magnitud a 1. Si el bit de signe el posem a zero, tindrem la representació de la magnitud positiva més gran que es pot representar. Si el bit de signe és 1, es tractarà de la magnitud negativa més gran que es pot representar. Aquests nombres, el més gran i el més petit, delimiten l'interval que conté els nombres que es poden representar, és a dir, el rang.

El nombre més gran representable en signe i magnitud i un format de coma fixa de 9 bits amb 3 de fraccionaris és el $011111,111_2 = +31,875_{(10)}$; el més petit és el $111111,111_2 = -31,875_{(10)}$. Per tant, el rang en decimal d'aquest format és:

$$[-31,875_{(10)}, +31,875_{(10)}]$$

En general, el nombre més gran que es pot representar amb n bits, dels quals m són fraccionaris, el podem calcular si suposem que tots el bits valen 1 i apliquem el TFN:

$$1 \cdot 2^{n-m-2} + 1 \cdot 2^{n-m-3} + \dots + 1 \cdot 2^0 + 1 \cdot 2^{-1} + \dots + 1 \cdot 2^{-m} = 2^{n-m-1} - 2^{-m}$$

en el qual hem aplicat la propietat següent:

$$111\dots11^k = 2^{k-1} + 2^{k-2} + \dots + 2^1 + 2^0 = \frac{2^k - 2^0}{2 - 1} = \frac{2^k - 1}{1} = 2^k - 1.$$

Exemple

Canvi de base 10 a base 2 del nombre $8,9453125_{(10)}$:

1. Part entera:

$$8 = 4 \cdot 2 + 0$$

$$4 = 2 \cdot 2 + 0$$

$$2 = 1 \cdot 2 + 0$$

$$1 = 0 \cdot 2 + 1$$

2. Part fraccionària:

$$0,9453125 \cdot 2 = 0,890625 + 1$$

$$0,890625 \cdot 2 = 0,781250 + 1$$

$$0,781250 \cdot 2 = 0,5625 + 1$$

$$0,5625 \cdot 2 = 0,125 + 1$$

$$0,125 \cdot 2 = 0,25 + 0$$

$$0,25 \cdot 2 = 0,5 + 0$$

$$0,5 \cdot 2 = 0 + 1$$

$$8,9453125_{(10)} = 1000,1111001_2$$

La precisió

La precisió és la distància entre dos nombres representables consecutius (vegeu el subapartat 2.1.2). Fàcilment podem comprovar que la precisió amb 3 bits fraccionaris és $0,001_2$ (distància entre el $0,000$ i el $0,001$).

La meitat d'aquest valor l'aconseguim desplaçant la coma una posició a l'esquerra (que equival a dividir per 2 en base 2). Per tant, la meitat de la precisió és $0,0001_2$.

El **rang** d'una representació en signe i magnitud i un format en coma fixa de n bits, en el qual m són fraccionaris, és

$$\left[-2^{n-m-1} + 2^{-m}, +2^{n-m-1} - 2^{-m} \right].$$

De la mateixa manera, el **rang** d'una representació de nombres fraccionaris sense signe en un format de coma fixa de n bits, en el qual m són fraccionaris, és el següent:

$$\left[0, +2^{n-m} - 2^{-m} \right].$$

Ampliació del nombre de bits d'un format de coma fixa

L'ampliació d'un format en coma fixa ha tenir en compte els possibles canvis en la posició de la coma. De fet, tan sols cal saber quants bits es destinen a l'ampliació de la part fraccionària i quants a l'ampliació de la part entera. Una ampliació de k bits de la part fraccionària comporta afegir k zeros a la dreta de la magnitud. Una ampliació de p bits de la part entera l'aconseguim si afegim p zeros a l'esquerra de la magnitud. Si treballem en signe i magnitud, haurem de separar el signe de la magnitud per tal de fer els canvis i després afegir-lo de nou a l'extrem esquerre.

L'**extensió** o ampliació de k bits per la part fraccionària i p bits per la part entera d'un format de coma fixa, tant en signe i magnitud com sense signe, l'aconseguim si afegim k zeros a la dreta de la magnitud i p zeros a l'esquerra de la magnitud.


Exemple

Per ampliar en 3 bits la part fraccionària i en 2 bits la part entera del nombre $111,001_{(SM2)}$ que està en coma fixa i signe i magnitud, afegirem 3 zeros a la dreta de la magnitud i 2 zeros a l'esquerra de la magnitud. La nova codificació en el cas de signe i magnitud és **10011,001000**_(SM2), en la qual es marquen en negre els dígits afegits.

Aquesta ampliació, en cas que el nombre $111,001_2$ fos una magnitud sense signe, donaria lloc a la codificació **00111,001000**₍₂₎.

Precisió d'un format de coma fixa

La precisió és la distància més petita entre dos nombres representables consecutius. Si treballem amb una representació en coma fixa de 3 bits, en la qual 1 és fraccionari i signe i magnitud, els nombres que podem representar són: $-1,1_{(2)}$, $-1,0_{(2)}$, $-0,1_{(2)}$, $0,0_{(2)}$, $+0,1_{(2)}$, $+1,0_{(2)}$ i $+1,1_{(2)}$. Com podem observar, tots ells estan separats en una distància $0,1_{(2)}$. Per aquest motiu, la precisió és $0,1_{(2)}$.

En les representacions de coma fixa, la precisió la determina el bit menys significatiu de la representació. 

La **precisió** d'una representació en coma fixa d' n bits, en la qual m són fraccionaris, és 2^{-m} .

Suma i resta en coma fixa


Les operacions de suma i de resta en coma fixa les fem a partir de l'algorisme habitual que hem descrit en l'apartat 1.6, com podem veure en els exemples següents:

$$\begin{array}{rcl}
 \begin{array}{r}
 1 \ 1 \ 0 \ 0 \\
 1 \ , \ 1 \ 0 \ 1 \\
 + \ 0 \ , \ 1 \ 0 \ 0 \\
 \hline
 1 \ 0 \ , \ 0 \ 0 \ 1
 \end{array} & \leftarrow \text{ròssecs} & \begin{array}{r}
 1 \ , \ 1 \ 0 \ 1 \\
 0 \ 0 \ 0 \\
 - \ 0 \ , \ 1 \ 0 \ 0 \\
 \hline
 1 \ , \ 0 \ 0 \ 1
 \end{array} \\
 & & \leftarrow \text{ròssecs} \\
 & & \leftarrow \text{resultat}
 \end{array}$$

Fixem-nos en el fet que un nombre fraccionari, com per exemple el $1,101_{(2)}$, el podem escriure de la forma $1101_{(2)} \cdot 2^{-3}$. Mitjançant aquest procediment, podem associar un nombre enter, el $1101_{(2)}$ en aquest cas, a un nombre fraccionari.

Si apliquem aquesta transformació als nombres $1,101_{(2)}$ i $0,100_{(2)}$ de la suma anteriors, obtenim: $1,101_{(2)} = 1101_{(2)} \cdot 2^{-3}$ i $0,100_{(2)} = 0100_{(2)} \cdot 2^{-3}$. La suma es pot fer de la forma següent:

$$1101_{(2)} \cdot 2^{-3} + 0100_{(2)} \cdot 2^{-3} = (1101_{(2)} + 0100_{(2)}) \cdot 2^{-3} = 10001_{(2)} \cdot 2^{-3}$$

Observem que mitjançant aquest procediment hem transformat una suma de nombres fraccionaris en una suma de nombres enters. 

Amb la transformació anterior, les operacions entre nombres fraccionaris les podem dur a terme a través d'operacions entre nombres enters.

Fixem-nos també en el fet que en la representació no apareix la ubicació de la coma, però, atès que tots els nombres amb els quals treballarem tindran la coma en la mateixa posició, no cal saber-ne la posició per a operar-hi.

És a dir, les operacions per a sumar els nombres $11,01_{(2)}$ i $00,01_{(2)}$ o els nombres $1,101_{(2)}$ i $0,001_{(2)}$ són idèntiques. La primera és:

$$11,01_{(2)} + 00,01_{(2)} = 1101_{(2)} \cdot 2^{-2} + 0001_{(2)} \cdot 2^{-2} = (1101_{(2)} + 0001_{(2)}) \cdot 2^{-2}$$

I la segona és:

$$1,101_{(2)} + 0,001_{(2)} = 1101_{(2)} \cdot 2^{-3} + 0001_{(2)} \cdot 2^{-3} = (1101_{(2)} + 0001_{(2)}) \cdot 2^{-3}$$

En tots dos casos, l'operació realitzada finalment és la suma dels dos nombres enters $1101_{(2)}$ i $0001_{(2)}$.

A continuació, sumem els nombres $11,111_{(2)}$ i $01,111_{(2)}$, que estan en un format de coma fixa de 5 bits, en el qual 3 són fraccionaris:

$$\begin{array}{r}
 1 \mid 1 \ 1 \ 1 \ 1 \quad \leftarrow \text{ròssecs} \\
 \mid 1 \ 1 \ 1 \ 1 \ 1 \\
 + \mid 0 \ 1 \ 1 \ 1 \ 1 \\
 \hline
 1 \mid 0 \ 1 \ 1 \ 1 \ 0 \quad \leftarrow \text{resultat}
 \end{array}$$

Per a representar el resultat ens cal un dígit més dels que tenim disponibles en el format definit. Per això, el resultat no és representable en el format especificat. S'ha produït sobreeximent, que podem reconèixer de la mateixa manera que en la suma de nombres naturals.

El sobreeximent en la suma de nombres sense signe en coma fixa el podem detectar de la mateixa manera que en el cas de la suma de nombres naturals. Recordem que la resta no pot donar lloc a sobreeximent.

El sobreeximent en la suma i la resta de nombres en coma fixa i signe i magnitud el podem detectar de la mateixa manera que en el cas de la suma i la resta de nombres enters representats en signe i magnitud.

La resta de nombres sense signe

La resta de nombres sense signe no pot donar lloc a sobreeximent perquè hem de restar la magnitud petita de la gran. Restar la magnitud gran de la magnitud petita no és una operació vàlida per a nombres sense signe, perquè el resultat hauria de ser un nombre amb signe.

Multiplicació i divisió per 2^k en coma fixa binària

Quan la base de numeració és 2 en un sistema posicional de base fixa, els pesos associats als dígits són potències de 2. Consegüentment, multiplicar per 2 es tradueix en augmentar en una unitat la potència de 2 associada a cada bit i dividir per 2 és equivalent a disminuir en una unitat la potència de 2 associada a cada bit.

Aplicant el TFN, podem escriure el nombre $0011,01_{(2)}$ de la manera següent:

$$0011,01_{(2)} = 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

Si multipliquem per 2, obtenim:

$$\begin{aligned}
 0011,01_{(2)} \cdot 2 &= (0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}) \cdot 2 = \\
 &= 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} = 00110,1_{(2)}
 \end{aligned}$$

En un format de coma fixa, la posició de la coma sempre és la mateixa i, per tant, per multiplicar o dividir per la base desplaçem els bits a l'esquerra o a la dreta i hi afegim els zeros necessaris.

! Vegeu la multiplicació i la divisió per potències de la base de numeració en el subapartat 1.8 d'aquest mòdul.

Multiplicar per 2^k un nombre en coma fixa sense signe equival a desplaçar els bits k posicions a l'esquerra i completar així els n bits del format amb l'addició a la dreta de k zeros.

Dividir per 2^k un nombre en coma fixa sense signe equival a desplaçar els bits k posicions a la dreta i completar així els n bits del format amb l'addició a l'esquerra de k zeros.

En coma fixa i 6 bits, dels quals 2 són fraccionaris, el resultat de multiplicar $0011,01_{(2)}$ per 2^2 l'obtenim si desplacem els bits 2 posicions a l'esquerra i afegim 2 zeros a la dreta:

$$0011,01_{(2)} \cdot 2 = 1101,00_{(2)}$$

Si el resultat no cap en el format, es produeix sobreiximent. Aquest succeeix quan es perden bits significatius (en coma fixa sense signe, bits a 1).


Es produeix **sobreiximent** en multiplicar un nombre sense signe en coma fixa per 2^k quan es perden un o més bits significatius (bits a 1) en desplaçar els bits k posicions.


La divisió per 2^k no produeix sobreiximent, però el quocient pot necessitar més bits fraccionaris que els disponibles en el format:

$$0101,00_{(2)} / 2^4 \approx 0000,01_{(2)} \quad (\text{en decimal, } 5_{(10)} / 2^4 \approx +0,25_{(10)})$$

El símbol \approx

El símbol \approx indica que es tracta d'una aproximació, no d'una igualtat.

La pèrdua de bits per la dreta equival a una aproximació per truncament. 

En coma fixa i signe i magnitud, les operacions de multiplicació i divisió per 2^k tenen les mateixes característiques que les descrites més amunt per coma fixa sense signe si separem el bit de signe. El bit de signe l'afegim en acabar l'operació. 

Activitats

27. Determineu quin valor decimal codifica la cadena de bits 1010010 en els supòsits següents:

- a) Si es tracta d'un nombre en coma fixa sense signe de 7 bits en el qual 4 són fraccionaris.
- b) Si es tracta d'un nombre en coma fixa sense signe de 7 bits en el qual 1 és fraccionari.

28. Codifiqueu els nombres $+12,85_{(10)}$, $+0,7578125_{(10)}$ i $-11,025_{(10)}$ en una representació fraccionària binària en signe i magnitud de 8 bits en la qual 3 són fraccionaris. Feu servir una aproximació per arrodoniment en cas que sigui necessari.

29. Si tenim una representació en coma fixa binària en signe i magnitud de 8 bits en la qual 3 bits són fraccionaris, determineu els nombres codificats per les cadenes de bits 01001111, 11001111, 01010100, 00000000 i 10000000.

30. Si les cadenes de bits 00101010, 11010010 i 10100010 representen nombres en coma fixa sense signe de 8 bits en els quals 3 són fraccionaris, representeu-los en un format de coma fixa sense signe de 12 bits en el qual 4 són fraccionaris.

31. Repetiu l'activitat anterior considerant que es tracta de nombres en signe i magnitud.

32. Determineu el rang de representació i la precisió en els formats següents:

- a) Coma fixa en signe i magnitud amb 8 bits dels quals 3 són fraccionaris.
- b) Coma fixa en signe i magnitud amb 8 bits dels quals 4 són fraccionaris.
- c) Coma fixa sense signe amb 8 bits dels quals 3 són fraccionaris.
- d) Coma fixa sense signe amb 8 bits dels quals 4 són fraccionaris.

33. Determineu la precisió necessària per a poder representar el nombre $+0,1875_{(10)}$ de manera exacta (sense error de representació) amb un format de coma fixa en base 2.

34. Determineu les característiques de rang i precisió, i també el nombre de dígitos enters i fraccionaris necessaris en un format de coma fixa en signe i magnitud, per poder representar de manera exacta els nombres $+31,875_{(10)}$ i $-16,21875_{(10)}$

35. Feu la suma i la resta dels parells de nombres següents, assumint que estan en coma fixa en signe i magnitud amb 8 bits, dels quals 3 són fraccionaris. Verifiqueu si el resultat és correcte:

- a) 00111000_2 i 10100000_2
- b) 10111010_2 i 11101100_2

3. Altres tipus de representacions

El funcionament dels computadors actuals es basa en l'electrònica digital, la característica distintiva de la qual és que tota la informació amb què treballa es codifica a partir de dos únics valors, que representem simbòlicament amb l'1 i el 0. Per tant, totes les dades que processa un computador digital han d'estar representades exclusivament per cadenes d'1 i 0, és a dir, per cadenes de bits. Llavors, el processament de les dades consisteix a aplicar operacions aritmètiques o lògiques a cadenes de bits.

En la secció precedent hem exposat les limitacions inherents a la tecnologia emprada en els computadors digitals actuals i les formes més usuals en què es codifiquen els valors numèrics. No ha estat una descripció exhaustiva de les formes de codificació de la informació numèrica, però sí una mostra representativa de la manera com la informació numèrica es codifica per a processar-la dins dels computadors digitals.

En els apartats següents mostrem, d'una banda, com codificar informació que inicialment no és numèrica fent servir, en darrer terme, els símbols 1 i 0; i, de l'altra, alguns sistemes de numeració alternatius que tenen especial interès en determinades circumstàncies.

3.1. Representació d'informació alfanumèrica

S'anomena *informació alfanumèrica* la informació no numèrica constituïda, bàsicament, pel conjunt de lletres, xifres i símbols que es fan servir a les descripcions textuais i que reben el nom genèric de *caràcters*.

El nombre de caràcters emprat en els textos és relativament gran: lletres majúscules, lletres minúscules, vocals accentuades, símbols de puntuació, símbols matemàtics, etc.

La representació dels caràcters es duu a terme assignant una cadena de bits única i específica a cada caràcter, és a dir, assignant a cada caràcter un codi binari. L'assignació de codis podria ser arbitrària, però a la pràctica és convenient seguir uns criteris que facilitin el processament de la informació codificada. Per exemple, una assignació de codis ascendent a les lletres de l'alfabet facilita l'ordenació alfabètica: el resultat d'una senzilla operació de resta entre els codis permetrà establir l'ordre alfabètic.

Seguint criteris que faciliten el tractament de les dades i amb la intenció de compatibilitzar la informació processada per sistemes diferents, s'han estan-

darditzat unes poques codificacions, d'entre les quals la codificació ASCII és la més estesa.

La codificació ASCII té una versió bàsica de 128 símbols que forma l'estàndard *de facto* que fan servir la majoria dels computadors i una versió estesa, no tan estandarditzada, que inclou 128 símbols més. En la versió estesa es fan servir 8 bits per a codificar els 256 símbols que inclou, mentre que en la bàsica se'n fan servir 7.

En la taula següent poden veure l'assignació de codis ASCII*. La representació numèrica associada a cada símbol l'obtenim a partir de les seves coordenades. L'índex de columna és el dígit decimal menys significatiu i el de fila, el més significatiu. Per exemple, el caràcter alfanumèric 3 es representa pel valor decimal $51_{(10)}$ (5u, d1), que en binari és el $00110011_{(2)}$.

*ASCII és la sigla de l'expressió anglesa *American standard code for information interchange*.

$\begin{smallmatrix} u \\ d \end{smallmatrix}$	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9
0u	NULL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1u	LF	VT	FF	CR	SO	SI	DLE	DC1	DC2	DC3
2u	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3u	RS	US	SP	!	"	#	\$	%	&	'
4u	()	*	+	,	-	.	/	0	1
5u	2	3	4	5	6	7	8	9	:	;
6u	<	=	>	?	@	A	B	C	D	E
7u	F	G	H	I	J	K	L	M	N	O
8u	P	Q	R	S	T	U	V	W	X	Y
9u	Z	[\]	^	_	`	a	b	c
10u	d	e	f	g	h	i	j	k	l	m
11u	n	o	p	q	r	s	t	u	v	w
12u	x	y	z	{		}	~	DEL		

Nota

És habitual fer servir la cursiva per a distingir els nombres (3, 25, 234) dels caràcters o cadenes de caràcters (3, 25, 234).

Els primers 31 codis i el darrer no corresponen a símbols del llenguatge o caràcters visibles (el caràcter 32 –identificat com a SP– representa l'espai en blanc). Aquests codis són caràcters de control utilitzats per a donar format al text o com a ordres per als dispositius perifèrics (terminals alfanumèrics o gràfics, impressores, etc.).

Cada cop és més habitual codificar text en més d'una llengua en els computadors. Per aquest motiu, s'ha anat popularitzant l'extensió de la codificació ASCII dels caràcters alfanumèrics a 2 bytes (16 bits), que utilitza l'estàndard anomenat *Unicode* i que inclou els caràcters de les grafies més importants.

Caràcters de control no visualitzables

Alguns caràcters de control no visualitzables són: DEL (esborrar), ESC (escapada), HT (tabulador horitzontal), LF (final de línia), CR (retorn a primera columna), FF (final de pàgina), STX (inici de text) o ETX (final de text).

Els codis ASCII de 8 bits dels caràcters visibles (no així els corresponents a caràcters de control) es poden convertir a Unicode si afegim 8 zeros a l'esquerra per completar els 16 bits de l'estàndard Unicode.

Exemple de processament de codis ASCII

Analitzem els codis ASCII per esbrinar com transformar el codi d'una lletra majúscula en l'equivalent en minúscula. Els codis consecutius a partir del codi 65 segueixen l'ordre de les lletres de l'alfabet anglès tant per a les majúscules com, a partir del codi 97, per a les minúscules. Per tant, la distància entre símbols de majúscules i minúscules és constant.

En concret, el caràcter *A* té el codi 65, mentre que el caràcter *a* té el 97. La diferència entre els codis és 32₍₁₀₎. Per tant, per a transformar el codi ASCII d'una lletra majúscula al codi ASCII de la mateixa lletra en minúscula, hem de sumar 32₍₁₀₎, al codi ASCII en binari.

L'estàndard Unicode

L'Unicode és estandarditzat per la ISO/IEC (*International Organization for Standardization / International Electrotechnical Commission*) amb l'identificador 10646.

Format Unicode

La codificació de textos en format Unicode és present a molts dels processadors de textos actuals com, per exemple, el Wordpad del Windows.

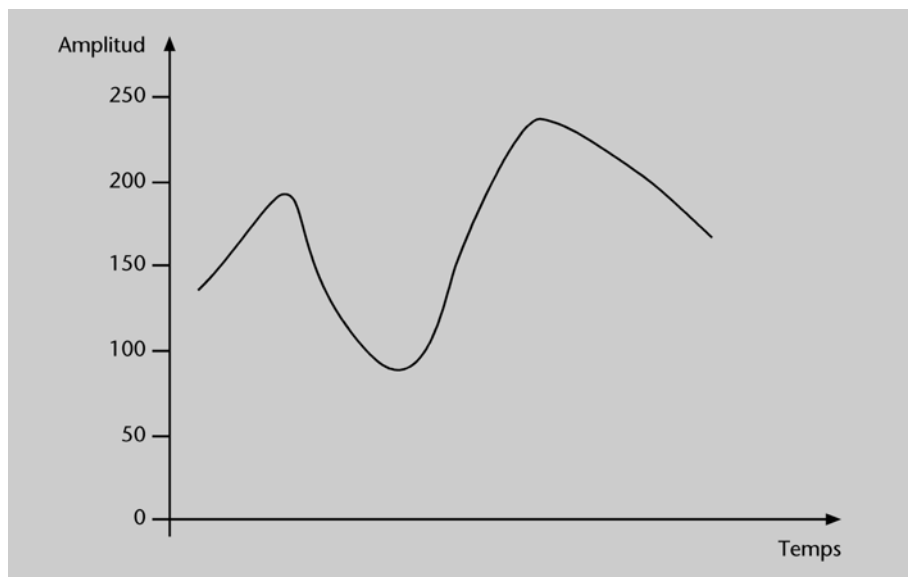
3.2. Codificació de senyals analògics

De vegades, les dades que ha de processar un computador provenen de dispositius que recullen informació de l'entorn. Un micròfon, per exemple, capta les ones sonores que hi arriben. Els sensors d'aquests dispositius són analògics, és a dir, generen un senyal elèctric de sortida que s'ajusta de manera contínua a la variació de l'estímul que reben. El resultat és un senyal elèctric, la variació en el temps del qual reflecteix la variació de l'estímul que ha arribat al sensor del dispositiu.

Un senyal elèctric analògic és aquell que codifica la informació mitjançant una variació contínua d'un paràmetre elèctric (tensió, freqüència, intensitat) que s'ajusta de manera proporcional a l'estímul original.

La figura 1 mostra un senyal analògic, en el qual l'amplitud varia de forma contínua al llarg del temps:

Figura 1



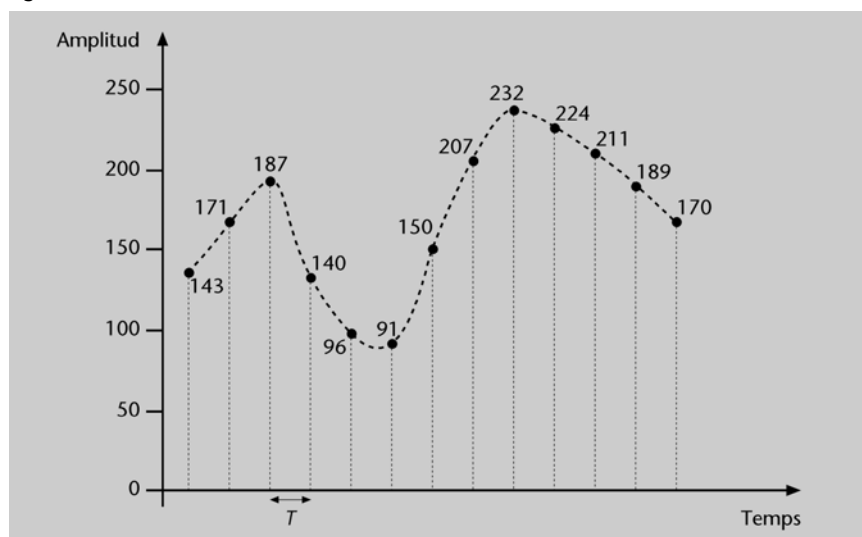
Els processadors digitals no poden tractar directament els senyals analògics i, com en el cas dels nombres o dels caràcters alfanumèrics, s'han de codificar fent servir únicament els símbols 1 i 0. El procés que permet aquesta transformació és la **digitalització**.

La **digitalització** consisteix a convertir una representació analògica en una representació digital binària, és a dir, en una seqüència ordenada de nombres binaris.

El procés de digitalització consta de tres etapes, que són el mostratge, la quantificació i la codificació binària:

1) El **mostratge** (o **discretització**) consisteix a prendre mostres del senyal analògic a intervals de temps regulars.

Figura 2

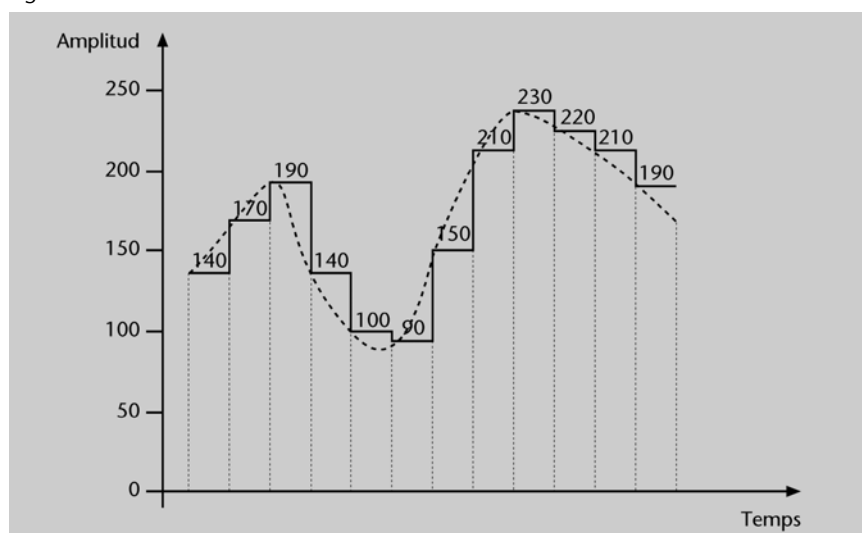


Mostratge d'un senyal analògic

Un mostratge del senyal dibuixat en la figura 1 a intervals de període T donaria el resultat que veiem en la figura 2.

2) La **quantificació** consisteix a assignar un valor, d'entre un conjunt finit, a l'amplitud del senyal en cada interval de mostratge.

Figura 3



Quantificació d'un senyal

Si admetem únicament valors múltiples de 10, la quantificació del mostreig de la figura 2 dóna lloc a la seqüència de valors numèrics: 140, 170, 190, 140, 100, etc., que representa una aproximació esglaonada al senyal continu original, com es mostra en la figura 3.

Com més petits siguin els intervals de temps del mostratge (fins a un cert límit més enllà del qual ja no guanyem res) i com més gran sigui el conjunt de valors admesos en la quantificació, més propera serà la informació digitalitzada a la informació analògica original.

3) La **codificació binària** consisteix a traduir els valors de les mostres a un sistema binari, és a dir, a expressar els valors mitjançant zeros i uns.

Exemple de codificació binària

Els valors de les amplituds que apareixen en la figura 3 van des del 90 fins al 230. Podem codificar aquest rang de valors decimals en binari fent servir 8 bits (ja que $2^7 < 230 < 2^8$). En la taula següent mostrem la codificació en binari dels valors decimals de la quantificació de la figura 3:

Amplitud	Codificació binària
90	01011010
100	01100100
140	10001100
150	10010110
170	10101010
190	10111110
210	11010010
220	11011100
230	11100110

De fet, convé tenir en compte que en la quantificació només s'han permès múltiples de 10. Així, en els valors d'amplitud podem menystenir el zero de la dreta i considerar el rang de valors [9, 23]. Per a codificar en binari els nombres dins d'aquest rang són suficients 5 bits (ja que $2^4 < 23 < 2^5$):

Amplitud	Codificació binària
90	01001
100	01010
140	01110
150	01111
170	10001
190	10011
210	10101
220	10110
230	10111

Aquesta codificació és més eficient, perquè fa servir menys bits. Si fem servir aquesta codificació, la informació que en la figura 3 s'expressava mitjançant una línia corba, ara s'expressa per la seqüència dels codis binaris següent:

01110
10001
10011

Avantatges de la digitalització

La digitalització és la tècnica que s'usa, per exemple, per a enregistrar música en un CD. El so es mostreja, es codifica en binari i es grava al CD fent-hi osques: un 1 es tradueix en fer una osca, un 0 es tradueix en no fer osca.

La tecnologia actual permet de fer el mostratge i la quantificació prou acurats perquè la distància entre els esglaons provocats per la digitalització no siguin perceptibles per l'oïda humana. D'altra banda, la digitalització evita els sorolls i distorsions que s'introdueixen amb mitjans analògics, cosa que fa que el so digital sigui de més qualitat que l'analògic.

01110
 01010
 01001
 01111
 10101
 10111
 10110
 10101
 10011


Aquesta seqüència de valors binaris constitueix una aproximació esglaonada a la corba contínua de la figura 1.

La digitalització permet transformar en nombres qualsevol senyal analògic del nostre entorn i aconseguir, així, que el puguem processar dins d'un computador digital.

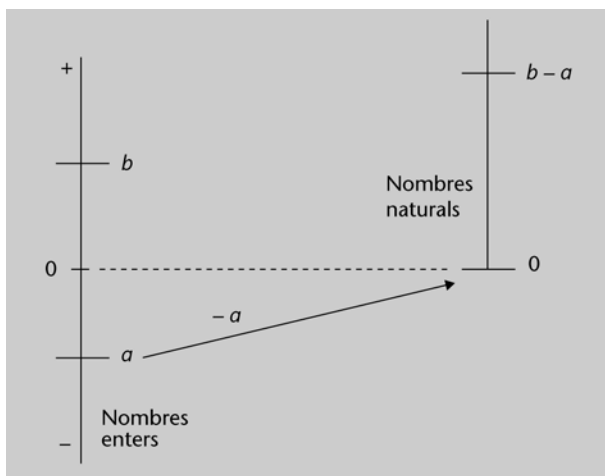
3.3. Altres representacions numèriques

La secció 2 l'hem dedicada a la descripció de les codificacions més usuals de nombres enters i fraccionaris tant amb signe com sense signe. Hi hem descrit les codificacions més utilitzades per a representar informació numèrica dins dels computadors però, amb tot, hi ha algunes representacions més que convé conèixer i que descrivim en els apartats següents.

3.3.1. Representació en excés a M

L'excés a M és un tipus de representació de nombres enters, en la qual l'estratègia que seguim és transformar el conjunt de valors numèrics enters que volem representar en un conjunt de nombres naturals, en el qual el valor més negatiu es codifica amb el zero. La resta de valors es codifiquen a partir del zero en ordre ascendent. 

En la figura següent mostrem gràficament aquesta estratègia:




Considerem l'interval $[-5, +5]$. Per desplaçar aquest rang de valors enters a un conjunt de valors naturals, tan sols hem de sumar 5 a cada nombre enter de l'interval. D'aquesta manera, els nombres passen a l'interval $[0, 10]$. Amb aquest desplaçament, el valor enter -5 dona lloc al valor natural 0, ja que $-5 - (-5) = -5 + 5 = 0$; el valor $+2$ dona lloc al valor natural 7, ja que $+2 - (-5) = +2 + 5 = 7$; etc.

Podem desplaçar l'interval $[a, b]$ de nombres enters a l'interval de nombres naturals $[0, b - a]$ si restem a a cada enter de l'interval $[a, b]$.

Aquest tipus d'estratègia l'anomenem representació en excés a M , en la qual M és el desplaçament que s'aplica a l'interval d'enters que volem codificar.

La **representació en excés a M** d'un nombre enter X l'obtenim quan sumem el desplaçament M al valor numèric X . Consegüentment, trobarem el valor d'un nombre codificat en excés a M si restem M a la codificació.

A tall d'exemple podem dir que, en una representació en excés a 10, el nombre enter -4 es representa mitjançant el nombre natural 6 (ja que $-4 + 10 = 6$) i que el 0 es representa mitjançant el nombre 10 (ja que $0 + 10 = 10$).

Atès que dins el computador la informació es codifica en binari, els nombres naturals emprats en la representació en excés a M es codifiquen en binari. 

Representació en excés a M

Per representar el valor $-6_{(10)}$ en excés a 7 i 4 bits, procedirem de la manera següent:

- 1) Sumem el desplaçament per trobar la codificació en decimal $-6_{(10)} + 7_{(10)} = 1_{(10)}$.
- 2) Codifiquem en binari i 4 bits el nombre obtingut $1_{(10)} = 0001_{(2)}$.

El valor $-6_{(10)}$ es codifica en excés a 7 i 4 bits com a 0001.

Per saber quin valor representa la cadena de bits 1100 que està codificada en excés a 7 i 4 bits, procedirem de la manera següent:

- 1) Fem un canvi de base per trobar la codificació en decimal $1100_{(2)} = 12_{(10)}$.
- 2) Restem el desplaçament per trobar el valor codificat $12_{(10)} - 7_{(10)} = +5_{(10)}$.

El valor que representa la cadena de bits 1100 codificada en excés a 7 i 4 bits és el $+5_{(10)}$.

L'excés a M és un tipus de representació de nombres enters emprada per a codificar el valor de l'exponent quan treballem en coma flotant.

3.3.2. Representació en coma flotant

Tot sovint hem de representar nombres molt grans (com ara la velocitat de transmissió de la llum en el buit, $c=299792500$ m/s) o bé nombres molt petits (com la massa d'un electró, $m_e = 0,000000000000000000000000091095$ kg) i potser de manera simultània. Per a evitar l'ús de molts dígitos en la representació d'aquests nombres, fem el format de **coma flotant**.

Coma flotant

L'avantatge de la coma flotant és la capacitat de representar amb pocs dígitos nombres que en altres formats necessiten molts dígitos per a representar-se.


Els nombres en **coma flotant** prenen la forma següent:

$$\pm R \cdot b^e$$

en la qual + o – indica el signe de la magnitud representada, R és un nombre fraccionari que rep el nom de **mantissa**, b és la base de numeració i e és un nombre enter que rep el nom d'**exponent**.

Terminologia

La **coma flotant** també rep el nom de **notació científica**.

La mantissa conté els dígitos significatius de la magnitud i està precedida pel signe d'aquesta magnitud. L'exponent indica el nombre de posicions a la dreta (exponent positiu) o a l'esquerra (exponent negatiu) que hem de desplaçar la coma fraccionària de la mantissa per obtenir el valor numèric representat. El nombre $+32,74_{(10)} \cdot 10^2$ és equivalent al $+3274_{(10)}$, mentre que $+32,74_{(10)} \cdot 10^{-1}$ és equivalent a $+3,274_{(10)}$. El valor de l'exponent indica la posició relativa de la coma fraccionària. 

Atenció

Al llarg del text farem servir les lletres S , e i R per a referir-nos, respectivament, al signe, l'exponent i la mantissa.

En el treball manual adaptem dinàmicament el nombre de dígitos emprats en la mantissa i l'exponent de la representació en coma flotant. Podem utilitzar $+2,995 \cdot 10^8$ o $+0,02995 \cdot 10^{10}$ segons convingui a les nostres necessitats. Dins dels computadors hem de cedir aquesta flexibilitat i adoptar restriccions que simplifiquin el processament de dades. Per això, assumim que la base de numeració és 2 i que el nombre de bits destinats a la mantissa i a l'exponent es fixa en l'especificació del format.

Símbol +

El símbol + no sol aparèixer davant de l'exponent o de la mantissa quan són positives.

Per a la representació en coma flotant dins dels computadors, assumim que la **base de numeració** és 2 i que la definició del format fixa el nombre de bits de la mantissa i el nombre de bits de l'exponent.

D'altra banda, la representació d'un valor numèric en coma flotant no és única. Per exemple, algunes representacions en coma flotant del nombre $26300_{(10)}$ són: $2,63_{(10)} \cdot 10^4$, $0,263_{(10)} \cdot 10^5$, $263_{(10)} \cdot 10^2$, $2630_{(10)} \cdot 10^1$, $26300_{(10)} \cdot 10^0$ o $263000_{(10)} \cdot 10^{-1}$. De nou, simplifiquem el tractament d'aquests nombres si restringim aquesta flexibilitat mitjançant la fixació del format de la mantissa. Per exemple, el format pot determinar que la coma és a la dreta del primer dígit no nul. Amb aquesta limitació, el $26300_{(10)}$ té una representació única que és $2,63 \cdot 10^4$.

Nota

En coma flotant, la representació d'un valor numèric no és única.

Nota

Fixant la posició de la coma de la mantissa es facilita la comparació de nombres.

Per evitar la multiplicitat de representacions d'un valor numèric, pròpia de la representació en coma flotant, en la definició del format **fixem la posició de la coma fraccionària respecte al primer dígit no nul de la mantissa**.

Quan la mantissa té fixada la posició de la coma fraccionària, rep el qualificatiu de **mantissa normalitzada**.

Les posicions més habituals en les quals es fixa la coma de la mantissa són l'esquerra del primer dígit no nul i, especialment, la dreta del primer dígit no nul. **!**

El valors numèrics representats en coma flotant amb mantissa normalitzada i coma a la dreta del primer bit no nul són de la forma següent:

$$\pm 1, x_{-1} x_{-2} \dots x_{-k} \cdot 2^e$$

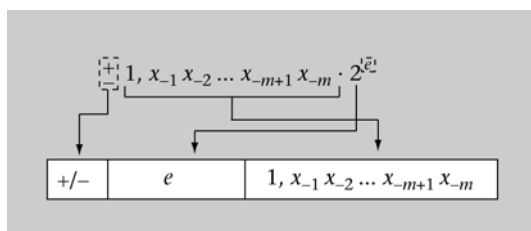
en el qual x_i són dígits binaris (bits), 2 és la base de numeració en decimal i e és l'exponent.

Coma flotant amb mantissa normalitzada

En les representacions en coma flotant amb la coma de la mantissa fixada a l'esquerra del primer dígit no nul, els nombres són de la forma següent:

$$\pm 0, 1x_{-2} \dots x_{-k} \cdot 2^e$$

La codificació en coma flotant ha d'incorporar la informació de signe (habitualment, 0 per als positius i 1 per als negatius), el valor de la mantissa i el de l'exponent. En canvi, no es guarda la base de numeració atès que s'assumeix que sempre és 2. En la figura següent mostrem l'ordre en què és usual disposar aquests valors:



El bit de signe

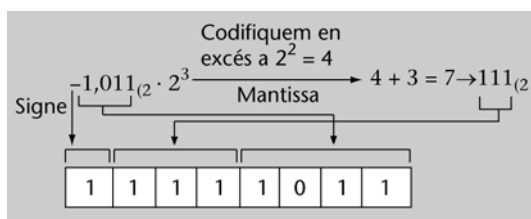
Com a norma, el bit de signe és 0 per a nombres positius i 1 per a nombres negatius.

Signe-exponent-mantissa

L'ordre de precedència signe-exponent-mantissa no és l'únic possible, però sí el més àmpliament utilitzat.

En coma flotant, l'exponent sol estar restringit als enters. Per a codificar-lo, el més habitual és fer servir excés a M , en el qual M pren freqüentment els valors 2^{q-1} o $2^{q-1} - 1$ en els quals q és el nombre de bits de l'exponent.

En coma flotant de 8 bits, mantissa normalitzada de 4 bits i exponent en excés a M , en el qual M pren el valor 2^{q-1} i q és el nombre de bits disponibles per a la representació de l'exponent, la codificació del nombre $-10,11_2 \cdot 2^2$ és la que mostrem en la figura següent:



Representació de l'exponent

Per a l'exponent, el més estès és l'ús d'excés a M , però es podria fer servir un altre tipus de codificació, com ara Ca2.

Nota

Dels 8 bits, 4 són per a la mantissa i 1 per al signe. En resten 3 per a l'exponent. Si M és 2^{q-1} , $M = 2^{3-1} = 2^2 = 4$. L'exponent es codifica en excés a 4.


En la taula següent mostrem alguns nombres representats en aquest format:

Nombres	S	e	R
$-1,101_2 \cdot 2^{-1}$	1	0 1 1	1 1 0 1
$1,101_2 \cdot 2^{-1}$	0	0 1 1	1 1 0 1

Nombres	S	e			R			
$1,0_{(2)} \cdot 2^{-3}$	0	0	0	1	1	0	0	0
$-1,10_{(2)} \cdot 2^{+1}$	1	1	0	1	1	1	0	0

En la taula precedent podem observar que el primer bit de la mantissa (columna R) sempre és 1, perquè es codifiquen de la forma $1, x_{-1}x_{-2} \dots$. Tenen una part fixa (1,) i una part variable ($x_{-1}x_{-2} \dots$). Per optimitzar recursos, podem emmagatzemar només la part variable, ja que la part fixa és coneguda i comú en tots els nombres. Això permetrà emmagatzemar 1 bit més per a la mantissa i augmentar-ne així la precisió, o bé utilitzar un bit menys en la representació. Les mantisses emmagatzemades mitjançant aquesta tècnica reben el nom de **mantisses amb bit implícit**.

La tècnica del **bit implícit** consisteix a emmagatzemar només la part variable de les mantisses normalitzades i assumir la part fixa com a coneguda i definida en el format de la representació.

L'ús del bit implícit permet emmagatzemar mantisses 1 bit més grans o reduir en 1 bit el nombre de bits necessaris per a la representació de la mantissa. 

Codificació d'un nombre decimal en coma flotant normalitzada i binària

Per codificar el nombre $+104_{(10)}$ en un format de coma flotant normalitzada de 8 bits, dels quals 3 es destinen a la mantissa amb bit implícit, i exponent en excés a M , seguirem el procés següent:

1) Codifiquem el nombre $+104_{(10)}$ en base 2.

Si apliquem el mètode de canvi de base basat en divisions successives, obtenim que $104_{(10)} = 1101000_{(2)}$.

2) Normalitzem la mantissa de la forma $1, x_{-1}x_{-2}x_{-3} \dots$: $1101000_{(2)} = 1,101 \cdot 2^6$.

3) Identifiquem el signe, l'exponent i la mantissa.

- a) El nombre és positiu; per tant, el bit de signe serà 0: $S = 0$.
- b) La mantissa d'aquest nombre és 1,101. El format indica que treballem amb una mantissa de 3 bits i bit implícit. Per tant, guardarem els 3 bits a la dreta de la coma: 101.
- c) L'exponent pren el valor 6.

4) Codifiquem l'exponent en excés a M . Dels 8 bits del format, 3 es fan servir per a la mantissa i 1 per al signe. Resten 4 per a l'exponent. Per tant, el valor de l'excés és $2^{4-1} = 2^3 = 8$. El $6_{(10)}$ codificat en excés a 8 és $6_{(10)} + 8_{(10)} = 14_{(10)}$. Si apliquem un altre cop el mètode de canvi de base basat en divisions successives, trobem que el $14_{(10)}$ en base 2 és el $1110_{(2)}$. Per tant, $e = 1110$.

5) Ajuntem les codificacions de signe, exponent i mantissa en l'ordre de precedència correcte ($S - e - R$) per tal d'obtenir la representació final:

S	exponent				mantissa		
0	1	1	1	0	1	0	1

Per tant, la codificació del nombre $+104_{(10)}$ en el format binari de coma flotant especificat és 01110101*.

Canvi de base

Per canviar a base 2 el $104_{(10)}$, fem divisions successives:

$$\begin{aligned}
 104 &= 52 \cdot 2 + 0 \\
 52 &= 26 \cdot 2 + 0 \\
 26 &= 13 \cdot 2 + 0 \\
 13 &= 6 \cdot 2 + 1 \\
 6 &= 3 \cdot 2 + 0 \\
 3 &= 1 \cdot 2 + 1 \\
 1 &= 0 \cdot 2 + 1
 \end{aligned}$$

$$104_{(10)} = 1101000_{(2)}$$

Canvi de base

Per canviar a base 2 el $14_{(10)}$, fem divisions successives:

$$\begin{aligned}
 14 &= 7 \cdot 2 + 0 \\
 7 &= 3 \cdot 2 + 1 \\
 3 &= 1 \cdot 2 + 1 \\
 1 &= 0 \cdot 2 + 1
 \end{aligned}$$

$$14_{(10)} = 1110_{(2)}$$

* El subratllat destaca la posició de l'exponent dins d'una cadena de bits que codifica un nombre en coma flotant.

Descodificació d'un nombre en coma flotant normalitzada binària

Per trobar el valor decimal del nombre 01010101 que està en un format de coma flotant normalitzada de 8 bits amb 4 bits de mantissa amb bit implícit i exponent en excés, seguirem el procés següent:

1) Identifiquem el signe.

Si assumim que el format manté el signe, l'exponent i la mantissa en aquest ordre, el bit de l'extrem esquerre codifica el signe. En el 01010101, el primer bit és 0; per tant, el signe és positiu.

2) Identifiquem la mantissa.

Atès que la mantissa ocupa les 4 posicions més baixes, es tracta dels bits 0101. Ara bé, el format indica l'existència de bit implícit. Per tant, la mantissa és en realitat 1,0101₍₂₎.

3) Identifiquem l'exponent.

L'exponent es determina pels 3 bits que resten:

S	Exponent			Mantissa			
0	1	0	1	0	1	0	1

M pren el valor 2^{q-1} , per la qual cosa $M = 2^{3-1} = 2^2 = 4$. Així, l'exponent codificat és $101_{(2)} - 4_{(10)} = 5_{(10)} - 4_{(10)} = 1_{(10)}$. L'exponent $e = 1_{(10)}$.

4) Ajuntem el signe, l'exponent i la mantissa.

Signe positiu, mantissa 1,0101₍₂₎ i exponent 1₍₁₀₎: el nombre representat és el $+1,0101 \cdot 2^1$.

5) Fem un canvi de base a fi de trobar el valor decimal.

Si apliquem el TFN, trobem que

$$+1,0101_{(2)} \cdot 2^1 = (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4}) \cdot 2^1 = 2^1 + 2^{-2} \cdot 2^1 + 2^{-4} \cdot 2^1 = +2,625_{(10)}$$

Per tant, el nombre 01010101₍₂₎ es codifica en el format de coma flotant especificat el valor decimal +2,625₍₁₀₎.

Per trobar el valor decimal del nombre 10010001₍₂₎ que està en un format de coma flotant normalitzada de 8 bits amb 5 bits de mantissa amb bit implícit i exponent en excés, seguirem el procés següent:

1) Identifiquem el signe.

Si assumim que el format manté el signe, l'exponent i la mantissa en aquest ordre, el bit de l'extrem esquerre codifica el signe. En el 10010001, el primer bit és 1; per tant, el signe és negatiu.

2) Identifiquem la mantissa.

Atès que la mantissa ocupa les 5 posicions més baixes, es tracta dels bits 10001. El format indica l'existència de bit implícit. Per tant, la mantissa és, en realitat, 1,10001₍₂₎.

3) Identifiquem l'exponent.

L'exponent és determinat pels 2 bits que resten:

S	e		Mantissa				
1	0	0	1	0	0	0	1

M pren el valor 2^{q-1} , per la qual cosa $M = 2^{2-1} = 2^1 = 2$. Així, l'exponent codificat és $00_{(2)} - 2_{(10)} = 0_{(10)} - 2_{(10)} = -2_{(10)}$. L'exponent $e = -2_{(10)}$.

4) Ajuntem el signe, l'exponent i la mantissa.

Signe negatiu, mantissa 1,10001₍₂₎ i exponent -2₍₁₀₎: el nombre representat és el $-1,10001_{(2)} \cdot 2^{-2}$.

5) Fem un canvi de base per trobar el valor decimal.

Si apliquem el TFN, trobem que

$$\begin{aligned} -1,10001_{(2)} \cdot 2^{-2} &= -(1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5}) \cdot 2^{-2} = \\ &= -(2^{-2} + 2^{-1} \cdot 2^{-2} + 2^{-5} \cdot 2^{-2}) = -0,3828125_{(10)} \end{aligned}$$

Per tant, el nombre 10010001₍₂₎ codifica en el format de coma flotant especificat el valor decimal -0,3828125₍₁₀₎.

3.3.3. Representació BCD

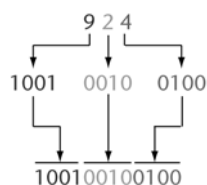
Una estratègia alternativa per a la representació de nombres és la codificació directa dels dígit decimal, sense fer un canvi de base. Com en el cas de la codificació de la informació alfanumèrica, hem d'assignar un codi binari a cada dígit decimal. En la taula següent mostrem la codificació binària dels dígit decimal en BCD:

Dígit decimal	Codificació binària
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

BCD


BCD és la sigla de *binary coded decimal*, és a dir, decimal codificat en binari.


Els codis BCD dels dígit decimal són de 4 bits. En la figura següent mostrem la manera en què podem representar un nombre decimal si codifiquem cada dígit individualment, anomenada **representació BCD**:



La **representació BCD** (*binary coded decimal*) consisteix a codificar els nombres decimals dígit a dígit. Cada dígit decimal se substitueix per 4 bits que corresponen a la codificació en binari del dígit decimal.

La codificació binària dígit a dígit dels nombres decimals aprofita parcialment la capacitat de representació. El codi 1100, per exemple, no es fa servir. De les 16 combinacions possibles que es poden fer amb 4 bits, només se'n fan servir 10. Conseqüentment, la representació d'un nombre en BCD necessita més bits que en binari.

Aquest tipus de representació és habitual en dispositius de sortida per a visualitzar dades. 

Que els díigits estiguin codificats en binari individualment no canvia el fet que es tracta de nombres decimals. Les operacions de suma i resta es desenvoluparan com en el cas de base 10. De fet, l'únic que s'està fent en aquesta forma de representació es canviar el símbol que utilitzem per a designar un díigit decimal per un codi binari que té la mateixa funció. Podem entendre-ho com un canvi de la simbologia per a representar els díigits. 

Activitats

36. Codifiqueu en BCD el nombre $125_{(10)}$.

37. Codifiqueu en BCD el nombre $637_{(10)}$.

38. Indiqueu quin nombre codifica la representació BCD 000100111000.

39. Codifiqueu el nombre $427_{(10)}$ en BCD i en binari. Compareu el nombre de bits necessaris en els dos casos.

40. Trobeu el valor decimal que codifiquen les cadenes de bits següents, interpretant que es tracta de nombres en un format de coma flotant de 8 bits amb mantissa normalitzada de la forma $1,X$ i amb bit implícit:

a) 11110010, en el qual la mantissa és de 4 bits.

b) 01010011, en el qual la mantissa és de 3 bits.

41. Feu les codificacions següents:

a) El nombre $-1,335_{(10)}$ en coma flotant de 8 bits, mantissa de 3 bits normalitzada de la forma $1,X$ i amb bit implícit emprant una aproximació per truncament.

b) Repetiu l'apartat anterior, però amb una aproximació per arrodoniment.

c) El nombre $10,0327_{(10)}$ en coma flotant de 9 bits, mantissa de 3 bits normalitzada de la forma $1,X$ i amb bit implícit emprant una aproximació per truncament.

42. Determineu si el nombre $2,89_{(10)} \cdot 10^{10}$ és representable en un format de coma flotant de 16 bits, amb mantissa normalitzada de la forma $1,X$, bit implícit i 5 bits per a l'exponent.

43. Determineu si el nombre $-1256_{(10)} \cdot 10^{-2}$ és representable en un format de coma flotant de 10 bits, amb mantissa normalitzada de la forma $1,X$, bit implícit i 6 bits per a l'exponent.

Resum

En aquest mòdul presentem una anàlisi dels sistemes de numeració posicionals i exposem les formes de representar valors numèrics que són habituals dins dels computadors. Els punts principals que tractem en aquest mòdul són els següents:

- El TFN i l'algorisme de divisions successives que permeten canviar entre bases diferents la representació d'un valor numèric.
- La representació de nombres naturals mitjançant representacions posicionals emprant base 2 (binari), base 16 (hexadecimal) i base 10 (decimal), i també les operacions de suma i resta de nombres naturals.
- Les limitacions derivades dels condicionaments físics dels computadors i les característiques que presenten els diferents formats de representació (rang i precisió), i també el fenomen del sobreiximent i les tècniques d'aproximació.
- La codificació dels nombres enters emprant les representacions en complement a 2 i signe i magnitud, i les operacions de suma i resta en cada una d'aquestes codificacions.
- La codificació de nombres fraccionaris amb i sense signe en coma fixa.
- L'empaquetament d'informació en hexadecimal i la codificació en BCD.

Exercicis d'autoavaluació

1. Codifiqueu en complement a 2 i signe i magnitud el nombre $-10_{(10)}$ emprant 8 bits.
2. Determineu el valor decimal que codifica la cadena de bits 00100100 en els supòsits següents:
 - a) Si es tracta d'un nombre codificat en complement a 2.
 - b) Si es tracta d'un nombre codificat en signe i magnitud.
3. Sumeu en binari els nombres 111010101100_2 i 11100010010_2 . Analitzeu el resultat obtingut.
4. Codifiqueu el nombre $+12,346_{(10)}$ en un format de coma fixa de 8 bits dels quals 3 són fraccionaris i signe i magnitud. Empreu una aproximació per truncament en cas que sigui necessari.
5. Codifiqueu el nombre $-45_{(10)}$ en un format de 8 bits i complement a 2.
6. Determineu el nombre mínim de bits enters i fraccionaris necessaris en coma fixa i signe i magnitud per codificar el nombre $-35,25_{(10)}$.
7. Codifiqueu els nombres $+12,25_{(10)}$ i el $+32,5_{(10)}$ en un format de coma fixa i signe i magnitud de 9 bits dels quals 2 són fraccionaris i sumeu-los.
8. Codifiqueu el nombre $178_{(10)}$ en la representació BCD.
9. Codifiqueu el nombre $+12,346_{(10)}$ en un format de coma flotant de 8 bits amb mantissa de 3 bits normalitzada de la forma $1,X$ amb bit implícit, i exponent en excés a M amb $M = 2^{q-1}$ (en el qual q és el nombre de bits de l'exponent). Empreu una aproximació per truncament en cas que sigui necessari.
10. Determineu el valor decimal que representa el codi $378_{(16)}$, sabent que es tracta d'un nombre en coma flotant de 12 bits amb mantissa de 4 bits normalitzada de la forma $1,X$ i bit implícit, exponent en excés a M amb $M = 2^{q-1}$ (en el qual q és el nombre de bits de l'exponent) i empaquetat en hexadecimal.

Solucionari

Activitats

1. Convertiu a base 10 els valors següents:

$$\text{a) } 10011101_{(2)} = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \\ = 128 + 16 + 8 + 4 + 1 = 157_{(10)}$$

$$\text{b) } 3AD_{(16)} = 3 \cdot 16^2 + A \cdot 16^1 + D \cdot 16^0 = 3 \cdot 16^2 + \underset{\substack{\uparrow \\ \text{Correspondència del dígit A en base 10}}}{10} \cdot 16^1 + \underset{\substack{\uparrow \\ \text{Dígit D en base 10}}}{13} \cdot 16^0 = 941_{(10)}$$

$$\text{c) } 333_{(4)} = 3 \cdot 4^2 + 3 \cdot 4^1 + 3 \cdot 4^0 = 48 + 12 + 3 = 63_{(10)}$$

$$\text{d) } 333_{(8)} = 3 \cdot 8^2 + 3 \cdot 8^1 + 3 \cdot 8^0 = 192 + 24 + 3 = 219_{(10)}$$

$$\text{e) } B2,3_{(16)} = B \cdot 16^1 + 2 \cdot 16^0 + 3 \cdot 16^{-1} = 11 \cdot 16^1 + 2 \cdot 16^0 + 3 \cdot 16^{-1} = \\ = 176 + 2 + 0,1875 = 178,1875_{(10)}$$

$$\text{f) } 3245_{(8)} = 3 \cdot 8^3 + 2 \cdot 8^2 + 4 \cdot 8^1 + 5 \cdot 8^0 = 1536 + 128 + 32 + 5 = 1701_{(10)}$$

$$\text{g) } AC3C_{(16)} = A \cdot 16^3 + C \cdot 16^2 + 3 \cdot 16^1 + C \cdot 16^0 = 10 \cdot 16^3 + 12 \cdot 16^2 + 3 \cdot 16^1 + 12 \cdot 16^0 = \\ = 40960 + 3072 + 48 + 12 = 44092_{(10)}$$

$$\text{h) } 1010,11_{(8)} = 1 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 0 \cdot 8^0 + 1 \cdot 8^{-1} + 1 \cdot 8^{-2} = \\ = 512 + 8 + 0,125 + 0,015625 = 520,140625_{(10)}$$

$$\text{i) } 110011,11_{(4)} = 1 \cdot 4^5 + 1 \cdot 4^4 + 0 \cdot 4^3 + 0 \cdot 4^2 + 1 \cdot 4^1 + 1 \cdot 4^0 + 1 \cdot 4^{-1} + 1 \cdot 4^{-2} = \\ = 1024 + 256 + 4 + 1 + 0,25 + 0,0625 = 1285,3125_{(10)}$$

$$\text{j) } 10011001,1101_{(2)} = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + \\ + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} = \\ = 128 + 16 + 8 + 1 + 0,5 + 0,25 + 0,0625 = 153,8125_{(10)}$$

$$\text{k) } 1110100,01101_{(2)} = 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + \\ + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} + 1 \cdot 2^{-5} = 64 + 32 + 16 + 4 + 0,25 + 0,125 + 0,03125 = \\ = 116,40625_{(10)}$$

2. Convertiu a base 2 els valors següents:

a) $425_{(10)}$

Dividim 425 per 2 successivament i registrem els residus de les divisions enteres. Aquests residus són els dígits binaris:

425	1	↑
212	0	
106	0	
53	1	
26	0	
13	1	
6	0	
3	1	
1		

Per tant: $425_{(10)} = 110101001_{(2)}$

b) $344_{(10)}$

344	0	↑
172	0	
86	0	
43	1	
21	1	
10	0	
5	1	
2	0	
1		

Per tant: $344_{(10)} = 101011000_{(2)}$

c) $31,125_{(10)}$

• Part fraccionària

$$\begin{array}{rcl} 0,125 \cdot 2 = 0,25 & = & \boxed{0} + 0,25 \\ 0,25 \cdot 2 = 0,50 & = & \boxed{0} + 0,50 \\ 0,50 \cdot 2 = 1,00 & = & \boxed{1} + 0,00 \end{array} \quad \downarrow$$

• Part entera

$$\begin{array}{r|l} 31 & 1 \\ 15 & 1 \\ 7 & 1 \\ 3 & 1 \\ 1 & \end{array} \quad \uparrow$$

Per tant, si $0,125_{(10)} = 0,001_{(2)}$ i $31_{(10)} = 11111_{(2)}$, llavors $31,125_{(10)} = 11111,001_{(2)}$

d) $4365,14_{(10)}$

• Part fraccionària

$$\begin{array}{rcl} 0,14 \cdot 2 = 0,28 & = & \boxed{0} + 0,28 \\ 0,28 \cdot 2 = 0,56 & = & \boxed{0} + 0,56 \\ 0,56 \cdot 2 = 1,12 & = & \boxed{1} + 0,12 \\ 0,12 \cdot 2 = 0,24 & = & \boxed{0} + 0,24 \\ 0,24 \cdot 2 = 0,48 & = & \boxed{0} + 0,48 \\ 0,48 \cdot 2 = 0,96 & = & \boxed{0} + 0,96 \\ 0,96 \cdot 2 = 1,92 & = & \boxed{1} + 0,92 \\ 0,92 \cdot 2 = 1,84 & = & \boxed{1} + 0,84 \\ 0,84 \cdot 2 = 1,68 & = & \boxed{1} + 0,68 \\ 0,68 \cdot 2 = 1,36 & = & \boxed{1} + 0,36 \\ 0,36 \cdot 2 = 0,72 & = & \boxed{0} + 0,72 \\ 0,72 \cdot 2 = 1,44 & = & \boxed{1} + 0,44 \end{array} \quad \downarrow$$

...

• Part entera

$$\begin{array}{r|l} 4365 & 1 \\ 2182 & 0 \\ 1091 & 1 \\ 545 & 1 \\ 272 & 0 \\ 136 & 0 \\ 68 & 0 \\ 34 & 0 \\ 17 & 1 \\ 8 & 0 \\ 4 & 0 \\ 2 & 0 \\ 1 & \end{array} \quad \uparrow$$

Per tant, si $0,14_{(10)} = 0,001000111101..._{(2)}$ i $4365_{(10)} = 1000100001101_{(2)}$, llavors

$$\begin{aligned} 4365,14_{(10)} &= 4365_{(10)} + 0,14_{(10)} = 1000100001101_{(2)} + 0,001000111101..._{(2)} = \\ &= 1000100001101,001000111101..._{(2)} \end{aligned}$$

3. Convertiu a hexadecimal els nombres següents:

a) $111010011,1110100111_{(2)}$

Base 2	0001	1101	0011,	1110	1001	1100
Base 16	1	D	3,	E	9	C

Per tant: $111010011,1110100111_{(2)} = 1D3,E9C_{(16)}$

b) $0,1101101_{(2)}$

Base 2	0,	1101	1010
Base 16	0,	D	A

Per tant: $0,1101101_{(2)} = 0,DA_{(16)}$

c) $45367_{(10)}$

Dividirem el valor numèric 45367 per 16 successivament i registrarem les restes de les divisions enteres realitzades. Aquests residus constitueixen els dígit hexadecimal:

$$\begin{array}{r|l} 45367 & 7 \\ 2835 & 3 \\ 177 & 1 \\ 11 & \end{array} \quad \uparrow$$

Per tant: $45367_{(10)} = B137_{(16)}$

d) $111011,1010010101_{(2)}$

Base 2	0011	1011,	1010	0101	0100
Base 16	3	B,	A	5	4

Per tant: $111011,1010010101_{(2)} = 3B,A54_{(16)}$

4. Convertiu els nombres hexadecimals següents a base 2, base 4 i base 8:

Podem aprofitar la propietat que $16 = 2^4$ i $16 = 4^2$ per a tractar el pas de base 16 a base 2 i a base 4 dígit a dígit. És a dir, cada dígit hexadecimal es transformarà en un conjunt de quatre dígits binaris, mentre que cada dígit hexadecimal es pot transformar en dos dígits en base 4.

El pas a base 8 no es pot fer directament des de base 16, atès que 16 no és potència de 8. Aprofitarem la base 2 com a base intermèdia. 8 és potència de 2 ($8 = 2^3$) i, per tant, tenim una correspondència directa: cada agrupació de tres dígits binaris es correspondrà amb un dígit octal.

a) $ABCD_{(16)}$

Base 16	A	B	C	D
Base 2	1010	1011	1100	1101
Base 4	22	23	30	31

Base 2	001	010	101	111	001	101
Base 8	1	2	5	7	1	5

Per tant: $ABCD_{(16)} = 1010101111001101_{(2)} = 22233031_{(4)} = 125715_{(8)}$ b) $45,45_{(16)}$

Base 16	4	5,	4	5
Base 2	0100	0101,	0100	0101
Base 4	10	11,	10	11

Base 2	001	000	101,	010	001	010
Base 8	1	0	5,	2	1	2

Per tant: $45,45_{(16)} = 1000101,01000101_{(2)} = 1011,1011_{(4)} = 105,212_{(8)}$ c) $96FF,FF_{(16)}$

Base 16	9	6	F	F,	F	F
Base 2	1001	0110	1111	1111,	1111	1111
Base 4	21	12	33	33,	33	33

Base 2	001	001	011	011	111	111,	111	111	110
Base 8	1	1	3	3	7	7,	7	7	6

Per tant: $96FF,FF_{(16)} = 1001011011111111,11111111_{(2)} = 21123333,3333_{(4)} = 113377,776_{(8)}$

5. Empleneu la taula següent:

Com podem veure, el valor numèric que en base 10 es representa per 74,3, amb una representació en base 2, 8 i 16 té un nombre infinit de dígits fraccionaris. En tots aquest casos obtenim una part fraccionària periòdica.

Binari	Octal	Hexadecimal	Decimal
1101100,110	154,6	6C,C	108,75

Binari	Octal	Hexadecimal	Decimal
11110010,010011	362,23	F2,4C	242,296875
10100001,00000011	241,006	A1,03	161,01171875
1001010,0100110011...	112,2314631463...	4A,4CCCCC...	74,3

6. Empaqueteu en hexadecimal la cadena de bits 10110001.

Per a empaquetar només cal agrupar els bits de 4 en 4 i fer-ne el canvi de base a hexadecimal. En aquest cas tenim els grups 1011 | 0001:

$$1011_{(2)} = B_{(16)}$$

$$0001_{(2)} = 1_{(16)}$$

Ara, agrupem tots aquests dígit en una única cadena i obtenim $10110001_{(2)} \rightarrow \mathbf{B1h}$.

7. Empaqueteu en hexadecimal el nombre $0100000111,111010_{(2)}$ que està en un format de coma fixa de 16 bits, dels quals 6 són fraccionaris.

Agrupem els bits de 4 en 4: 0100 | 0001 | 1111 | 1010 i fem el canvi de base de cada grup:

$$0100_{(2)} = 4_{(16)}$$

$$0001_{(2)} = 1_{(16)}$$

$$1111_{(2)} = F_{(16)}$$

$$1010_{(2)} = A_{(16)}$$

Per tant, la cadena 0100000111111010 s'empaqueta en hexadecimal per **41FAh**.

Observeu que no hem fet un canvi de base del nombre fraccionari representat per la cadena de bits original, sinó que hem empaquetat els bits sense tenir en compte el seu sentit.

8. Desempaqueteu la cadena de bits A83h

Obtenim la codificació binària de cada dígit:

$$A_{(16)} = 1010_{(2)}$$

$$8_{(16)} = 1000_{(2)}$$

$$3_{(16)} = 0011_{(2)}$$

Tenint en compte aquesta correspondència, A83h codifica la cadena de bits 101010000011.

a) Trobeu el valor decimal si es tracta d'un nombre natural.

Interpretat així, es tracta del nombre binari $101010000011_{(2)}$. Trobarem el valor decimal si apliquem el TFN:

$$101010000011 = 2^{11} + 2^9 + 2^7 + 2^1 + 2^0 = 2048 + 512 + 128 + 2 + 1 = \mathbf{2691}_{(10)}.$$

b) Trobeu el valor decimal si es tracta d'un nombre en coma fixa sense signe de 12 bits, dels quals 4 són fraccionaris.

Amb aquesta interpretació, es tracta del nombre binari $10101000,0011_{(2)}$. Trobarem el valor decimal aplicant el TFN:

$$10101000,0011 = 2^7 + 2^5 + 2^3 + 2^{-3} + 2^{-4} = 128 + 32 + 8 + 0,125 + 0,0625 = \mathbf{168,1875}_{(10)}.$$

9. Considerem el nombre $1010,101_{(2)}$.

a) Feu el canvi a base 16.

Per fer el canvi a base 16 hem de fer agrupacions de 4 bits a partir de la coma fraccionària:

Base 2	1010,	1010
Base 16	A,	A

Per tant, el nombre $1010,101_{(2)}$ en hexadecimal és $A,A_{(16)}$.

b) Feu l'empaquetament hexadecimal.

Per fer l'empaquetament, no hem de tenir en compte la posició de la coma fraccionària:

Base 2	0101	0,101
Base 16	5	5

Per tant, el nombre $1010,101_2$ s'empaqueta en hexadecimal com 55h.

10. Feu les operacions següents en la base especificada:

a.

$$\begin{array}{r}
 1 \ 1 \ 1 \ 1 \ 1 \\
 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \quad (2) \\
 + \quad 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \quad (2) \\
 \hline
 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \quad (2)
 \end{array}$$

d.

$$\begin{array}{r}
 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \quad (2) \\
 1 \quad \quad 1 \\
 - \quad 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \quad (2) \\
 \hline
 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 0 \quad (2)
 \end{array}$$

b.

$$\begin{array}{r}
 2 \ 3 \ 4 \ 5 \quad (8) \\
 + \quad \quad 3 \ 2 \ 1 \quad (8) \\
 \hline
 2 \ 6 \ 6 \ 6 \quad (8)
 \end{array}$$

e.

$$\begin{array}{r}
 2 \ 3 \ 4 \ 5 \quad (8) \\
 - \quad \quad 3 \ 2 \ 1 \quad (8) \\
 \hline
 2 \ 0 \ 2 \ 4 \quad (8)
 \end{array}$$

c.

$$\begin{array}{r}
 1 \\
 A \ 2 \ 3 \ F \quad (16) \\
 + \quad 5 \ 4 \ A \ 3 \quad (16) \\
 \hline
 F \ 6 \ E \ 2 \quad (16)
 \end{array}$$

f.

$$\begin{array}{r}
 A \ 2 \ 3 \ F \quad (16) \\
 1 \ 1 \\
 - \quad 5 \ 4 \ A \ 3 \quad (16) \\
 \hline
 4 \ D \ 9 \ C \quad (16)
 \end{array}$$

11. Feu les operacions següents en la base especificada:

a.

$$\begin{array}{r}
 1 \ 1 \\
 6 \ 2, \ 4 \ 8 \quad (16) \\
 + \quad 3 \ 5, \ D \ F \quad (16) \\
 \hline
 9 \ 8, \ 2 \ 7 \quad (16)
 \end{array}$$

b.

$$\begin{array}{r}
 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \quad 1 \ 1 \ 1 \\
 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1, \ 1 \ 1 \ 0 \ 1 \ 1 \quad (2) \\
 + \quad 1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0, \ 1 \ 1 \ 1 \quad (2) \\
 \hline
 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0, \ 1 \ 0 \ 1 \ 1 \ 1 \quad (2)
 \end{array}$$

c.						d.															
	6	2,	4	8	(16		1	1	1	1	0	1	1	0	1,	1	1	0	1	1	(2
	1	1	1						1	1					1	1	1				
–	3	5,	D	F	(16	–	1	0	0	1	1	0	1	0	0,	1	1	1			(2
	2	C,	6	9	(16		0	1	0	1	1	1	0	0	0,	1	1	1	1	1	(2

12. Feu les multiplicacions següents:

La multiplicació d'un nombre per b^k , en el qual b és la base de numeració, equival a desplaçar la coma fraccionària k posicions a la dreta.

a) $128,7_{(10)} \cdot 10^4 = 128,7_{(10)} \cdot 10000_{(10)} = 1287000_{(10)}$

b) $AFD_{(16)} \cdot 16^2 = AFD_{(16)} \cdot 100_{(16)} = AFD00_{(16)}$

c) $1101,01_{(2)} \cdot 2^2 = 1101,01_{(2)} \cdot 100_{(2)} = 110101_{(2)}$

13. Trobeu el quocient i el residu de les divisions enteres següents:

La divisió d'un nombre per b^k , en el qual b és la base de numeració, equival a desplaçar la coma fraccionària k posicions a l'esquerra.

a) $52978_{(10)} / 10^3 = 52978_{(10)} / 1000_{(10)} = 52,978_{(10)}$

El quocient de la divisió entera és $52_{(10)}$. La resta és $978_{(10)}$.

b) $3456_{(16)} / 16^2 = 3456_{(16)} / 100_{(16)} = 34,56_{(16)}$

El quocient de la divisió entera és $34_{(16)}$. La resta és $56_{(16)}$.

c) $100101001001_{(2)} / 2^8 = 100101001001_{(2)} / 100000000_{(2)} = 1001,01001001_{(2)}$

El quocient de la divisió entera és $1001_{(2)}$. La resta és $01001001_{(2)}$.

14. Determineu el rang i la precisió dels formats de coma fixa sense signe $x_1x_0,x_{-1}x_{-2}x_{-3}$ i $x_2x_1x_0,x_{-1}x_{-2}$, en els quals x_i és un dígit decimal.

Atès que la base de numeració és 10, el rang de la representació del format $x_1x_0,x_{-1}x_{-2}x_{-3}$ és $[0, 99,999_{(10)}]$. La precisió d'aquest format és $0,001_{(10)}$ perquè aquesta és la distància entre dos nombres consecutius representables en aquest format, com ara el $12,121_{(10)}$ i el $12,122_{(10)}$.

De manera similar, el rang de representació del format $x_2x_1x_0,x_{-1}x_{-2}$ és $[0, 999,99_{(10)}]$, i la seva precisió és $0,01_{(10)}$, la distància entre dos nombres consecutius representables en el format, com ara el $45,77_{(10)}$ i el $45,78_{(10)}$.

15. Determineu si el nombre $925,4$ es pot representar en els formats indicats en l'activitat 14.

El nombre $925,4_{(10)}$ no el podem representar en el format $x_1x_0,x_{-1}x_{-2}x_{-3}$, ja que aquest nombre és fora del rang de representació. En canvi, el podem representar en el format $x_2x_1x_0,x_{-1}x_{-2}$, atès que es troba dins del rang $[0, 999,99_{(10)}]$. A més, el podem representar de manera exacta, perquè en el format hi ha disponibles dígit fraccionaris suficients.

16. Representeu en el format de coma fixa i sense signe $x_1x_0,x_{-1}x_{-2}$, en el qual x_i és un dígit decimal, els nombres següents:

Per tal d'escriure aquests nombres en el format indicat, cal escriure'ls amb dos dígit enters i dos dígit fraccionaris:

a) $10_{(10)}$ s'escriu $10,00$

b) $10,02_{(10)}$ s'escriu $10,02$

c) $03,1_{(10)}$ s'escriu $03,10$

d) $03,2_{(10)}$ s'escriu $03,20$

17. Determineu la quantitat de nombres que podem representar en el format $x_2x_1x_0,x_{-1}x_{-2}x_{-3}$, en el qual x_i és un dígit decimal.

La màxima quantitat de nombres que podem representar en un format és b^k , en el qual k és el nombre de dígit disponibles en el format i b la base de numeració. Atès que es tracta d'un

format decimal, cada dígit pot prendre 10 valors diferents (0 – 9) i, com que el format disposa de 6 dígits per a la representació, podem representar un total de 10^6 nombres. Fixem-nos que la quantitat de nombres que podem representar no depèn de la posició de la coma.

18. Calculeu l'error de representació que es comet quan representem en el format $x_2x_1x_0, x_{-1}x_{-2}$, en el qual x_i és un dígit decimal, els nombres següents:

a) $223,45_{(10)}$

El nombre $223,45_{(10)}$ està directament representat en el format $x_2x_1x_0, x_{-1}x_{-2}$. Per tant, es tracta d'un nombre representable i l'error comès és zero.

b) $45,89_{(10)}$

El nombre $45,89_{(10)}$ és representable directament en el format $x_2x_1x_0, x_{-1}x_{-2}$. La representació és $045,89_{(10)}$ i és exacta. Per tant, l'error de representació és zero.

c) $55,6356_{(10)}$

El nombre $55,6356_{(10)}$ no es pot representar directament en el format $x_2x_1x_0, x_{-1}x_{-2}$, atès que té 4 dígits fraccionaris. Haurem de fer una aproximació, cosa que comporta un cert error d'aproximació.

Amb una aproximació per truncament, la representació serà $055,63_{(10)}$. L'error de representació que es comet és $|55,6356_{(10)} - 55,63_{(10)}| = 0,0056_{(10)}$.

Per trobar la representació una aproximació per arrodoniment, hem de sumar la meitat de la precisió i truncar el resultat a 2 dígits fraccionaris: $55,6356_{(10)} + 0,005_{(10)} = 55,6406_{(10)}$, que amb el truncament a 2 dígits fraccionaris queda $55,64_{(10)}$. L'error de representació comès és, en aquest cas, $|55,6356_{(10)} - 55,64_{(10)}| = 0,0044_{(10)}$.

d) $23,56_{(10)}$

El nombre $23,56_{(10)}$ és representable directament en el format $x_2x_1x_0, x_{-1}x_{-2}$. La representació és $023,56_{(10)}$ i és exacta. Per tant, l'error de representació és zero.

19. Escolliu el format hexadecimal que faci servir el mínim nombre de dígits i que permeti representar el nombre $16,25_{(10)}$ de manera exacta. Quin és el rang i la precisió del format?

Per representar aquest nombre en hexadecimal, primer el passarem a binari. La representació binària de $16,25_{(10)}$ és $10000,01_{(2)}$.

Per a la representació hexadecimal d'aquest nombre cal afegir zeros als extrems fins a disposar de grups de 4 bits complets a banda i banda de la coma decimal. Llavors tenim:

$$00010000,0100_{(2)} = 10,4_{(16)}.$$

- El rang d'aquesta representació és: $[0 (00,0_{(16)}), 255,9375_{(10)} (FF,F_{(16)})]$.
- La seva precisió és: $0,0625_{(10)} = 00,1_{(16)} - 00,0_{(16)}$.

20. Quin és el nombre més petit que cal sumar a $8341_{(10)}$ perquè es produeixi sobreiximent en una representació decimal (base 10) de quatre dígits?

Per tal que es produeixi sobreiximent en una representació decimal de 4 dígits sense signe, hem de sobrepassar el nombre $9999_{(10)}$; per tant, tindrem sobreiximent quan la suma de dos nombres sigui $10000_{(10)}$. Llavors, el nombre més petit que hem de sumar a $8341_{(10)}$ és $10000_{(10)} - 8341_{(10)} = 1659_{(10)}$.

21. Convertiu els valors decimals següents a binaris en els sistemes de representació de signe i magnitud i complement a 2, amb un format enter de 8 bits:

Passem els valors numèrics a binari:

a) 53

53	1
26	0
13	1
6	0
3	1
1	

b) -25

25	1
12	0
6	0
3	1
1	

c) 93

93	1
46	0
23	1
11	1
5	1
2	0
1	

$$+53_{(10)} = +110101_{(2)}$$

$$-25_{(10)} = -11001_{(2)}$$

$$+93_{(10)} = +1011101_{(2)}$$

d) -1

1	1
0	

e) -127

127	1
63	1
31	1
15	1
7	1
3	1
1	

f) -64

64	0
32	0
16	0
8	0
4	0
2	0
1	

$$-1_{(10)} = -1_{(2)}$$

$$-127_{(10)} = -1111111_{(2)}$$

$$-64_{(10)} = -1000000_{(2)}$$

Per obtenir la representació en signe i magnitud, tan sols hem de posar el bit de signe i afegir la magnitud expressada en 7 bits:

Base 10	Base 2	Signe i magnitud
+53 ₍₁₀₎	+110101 ₍₂₎	00110101 _(SM2)
-25 ₍₁₀₎	-11001 ₍₂₎	10011001 _(SM2)
+93 ₍₁₀₎	+1011101 ₍₂₎	01011101 _(SM2)
-1 ₍₁₀₎	-1 ₍₂₎	10000001 _(SM2)
-127 ₍₁₀₎	-1111111 ₍₂₎	11111111 _(SM2)
-64 ₍₁₀₎	-1000000 ₍₂₎	11000000 _(SM2)

La representació en Ca2 de les magnituds positives coincideix amb la representació en signe i magnitud. La representació en Ca2 de les magnituds negatives es pot obtenir de diverses maneres:

- Podem fer l'operació $2^8 - |X|$ en base 10 i passar posteriorment el resultat a binari.
- Podem fer l'operació $2^8 - |X|$ directament en base 2.
- Apliquem un canvi de signe a la magnitud positiva en Ca2.

a) El +53₍₁₀₎ = +110101₍₂₎ es representa per 00110101_(Ca2) en Ca2.

b) Podem obtenir la representació en Ca2 del -25₍₁₀₎ de la manera següent:

- $2^8 - 25 = 256_{(10)} - 25_{(10)} = 231_{(10)} = 11100111_{(Ca2)}$, o bé,
- $2^8 - 25 = 100000000_{(2)} - 11001_{(2)} = 11100111_{(Ca2)}$, o bé,
- +25₍₁₀₎ = +11001₍₂₎ → Representació de la magnitud positiva → 00011001_(Ca2) →
→ canvi de signe → 11100111_(Ca2)

c) El +93₍₁₀₎ = +1011101₍₂₎ es representa per 01011101_(Ca2) en Ca2.

d) Obtenim la representació en Ca2 del -1₍₁₀₎:

- $2^8 - 1 = 256_{(10)} - 1_{(10)} = 255_{(10)} = 11111111_{(Ca2)}$, o bé,
- $2^8 - 1 = 100000000_{(2)} - 1_{(2)} = 11111111_{(Ca2)}$, o bé,
- +1₍₁₀₎ = +1₍₂₎ → Representació de la magnitud positiva → 00000001_(Ca2) →
→ Canvi de signe → 11111111_(Ca2)

e) La representació en Ca2 del -127₍₁₀₎ la podem obtenir:

- $2^8 - 127 = 256_{(10)} - 127_{(10)} = 129_{(10)} = 10000001_{(Ca2)}$, o bé,
- $2^8 - 127 = 100000000_{(2)} - 1111111_{(2)} = 10000001_{(Ca2)}$, o bé,
- +127₍₁₀₎ = +1111111₍₂₎ → Representació de la magnitud positiva → 01111111_(Ca2) →
→ Canvi de signe → 10000001_(Ca2)

f) La representació en Ca2 del -64₍₁₀₎ l'obtenim:

- $2^8 - 64 = 256_{(10)} - 64_{(10)} = 192_{(10)} = 11000000_{(Ca2)}$, o bé,
- $2^8 - 64 = 100000000_{(2)} - 1000000_{(2)} = 11000000_{(Ca2)}$, o bé,
- +64₍₁₀₎ = +1000000₍₂₎ → Representació de la magnitud positiva → 01000000_(Ca2) →
→ Canvi de signe → 11000000_(Ca2)

• Resta $A - B$

A és negatiu i B també. Per tant, restarem la magnitud petita (B) de la magnitud gran (A) i aplicarem el signe de la magnitud gran (A) al resultat (no es pot produir sobreiximent):

Resta $A - B$

$ \begin{array}{r} 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1 \\ 1\ 1 \\ -\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0 \end{array} $	$ \begin{array}{r} 2\ 9\ 5\ (10) \\ 1 \\ -\ 2\ 9\ (10) \\ \hline 2\ 6\ 6\ (10) \end{array} $
--	--

$A - B = 1100001010_2 = -266_{(10)}$

• Suma $A + C$

A és negatiu i C és positiu. Per tant, restarem la magnitud petita (A) de la magnitud gran (C) i aplicarem el signe de la magnitud gran (C) al resultat (no es pot produir sobreiximent):

Suma $A + C$

$ \begin{array}{r} 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ 1\ 1 \\ -\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1 \\ \hline 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0 \end{array} $	$ \begin{array}{r} 3\ 4\ 7\ (10) \\ 1 \\ -\ 2\ 9\ 5\ (10) \\ \hline 0\ 5\ 2\ (10) \end{array} $
--	---

$A + C = 0000110100_2 = +52_{(10)}$

• Resta $A - C$

A és negatiu i C és positiu. Per tant, farem la suma de les magnituds (sense signe) i, si no hi ha sobreiximent, afegirem un 1 com a bit de signe al resultat obtingut, per obtenir el resultat en la representació en signe i magnitud:

Resta $A - C$

Sobreiximent
↓

$ \begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1 \\ +\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0 \end{array} $	$ \begin{array}{r} 1\ 1 \\ 2\ 9\ 5\ (10) \\ +\ 3\ 4\ 7\ (10) \\ \hline 6\ 4\ 2\ (10) \end{array} $
--	--

$A - C = -642_{(10)}$
 No es pot representar amb 10 bits en el format de signe i magnitud.

- Resta $B - C$

B és negatiu i C és positiu. Per tant, farem la suma de les magnituds (sense signe) i, si no hi ha sobreiximent, afegirem un 1 com a bit de signe al resultat obtingut, per obtenir el resultat en la representació en signe i magnitud:

Resta $B - C$

$ \begin{array}{r} 1 \ 1 \ 1 \ 1 \ 1 \\ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \\ + \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \\ \hline 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \end{array} $	$ \begin{array}{r} 1 \\ 2 \ 9 \ (10) \\ + \ 3 \ 4 \ 7 \ (10) \\ \hline 3 \ 7 \ 6 \ (10) \end{array} $
--	---

$B - C = 1101111000_2 = -376_{(10)}$

- Suma $B + C$

B és negatiu i C és positiu. Per tant, restarem la magnitud petita (B) de la magnitud gran (C) i aplicarem el signe de la magnitud més gran (C) al resultat (no es pot produir sobreiximent):

Suma $B + C$

$ \begin{array}{r} 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \\ 1 \ 1 \ 1 \ 1 \\ - \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \end{array} $	$ \begin{array}{r} 3 \ 4 \ 7 \ (10) \\ 1 \\ - \ 2 \ 9 \ (10) \\ \hline 3 \ 1 \ 8 \ (10) \end{array} $
--	---

$B + C = 0100111110_2 = +318_{(10)}$

25. Repetiu l'activitat anterior considerant que les cadenes representen nombres en complement a 2.

Per fer les operacions de suma en Ca_2 , operarem directament sobre les representacions. El resultat serà correcte sempre que no es produeixi sobreiximent. Per fer les operacions de resta, aplicarem un canvi de signe al subtrahend i farem una operació de suma:

$$\begin{aligned}
 A &= 1100100111 = -217_{(10)} \\
 B &= 1000011101 = -483_{(10)} \\
 C &= 0101011011 = +347_{(10)}
 \end{aligned}$$

- Suma $A + B$

Suma $A + B$

Sobreiximent
↓

$ \begin{array}{r} 1 \qquad \qquad \qquad 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \\ 1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\ + \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \end{array} $	$ \begin{array}{r} 1 \ 1 \\ -2 \ 1 \ 7 \ (10) \\ + \ -4 \ 8 \ 3 \ (10) \\ \hline -7 \ 0 \ 0 \ (10) \end{array} $
---	--

$A + B = -700_{(10)}$

En aplicar un canvi de signe al 0101011011 obtenim el 1010100101, que serà el valor que utilitzarem a la suma:

Suma $A + (-C)$	
Sobreeiximent	
↓	
1 1 1 1 1 1	
1 1 0 0 1 0 0 1 1 1 (Ca2)	-2 1 7 (10)
+ 1 0 1 0 1 0 0 1 0 1 (Ca2)	+ -3 4 7 (10)
1 0 1 1 1 0 0 1 1 0 0 (Ca2)	-5 6 4 (10)
$A - C = -564_{(10)}$	

Es produeix sobreeiximent, atès que en sumar dos nombres negatius, n'obtenim un de positiu. El resultat no cap en el format de sortida. Recordem que el rang de representació d'enters amb 10 bits amb el format de complement a 2 és $[-2^{10-1}, 2^{10-1} - 1] = [-512, 511]$.

- Resta $B - C$

Aplicarem un canvi de signe a C i farem una operació de suma. En aplicar un canvi de signe al 0101011011, obtenim el 1010100101, que serà el valor que utilitzarem a la suma:

Suma $B + (-C)$	
Sobreeiximent	
↓	
1 1 1 1 1 1 1 1	
1 0 0 0 0 1 1 1 0 1 (Ca2)	-4 8 3 (10)
+ 1 0 1 0 1 0 0 1 0 1 (Ca2)	+ -3 4 7 (10)
1 0 0 1 1 0 0 0 1 0 (Ca2)	-8 3 0 (10)
$B - C = -830_{(10)}$	

Es produeix sobreeiximent, atès que en sumar dos nombres negatius, n'obtenim un de positiu. El resultat no cap al format de sortida.

- Suma $B + C$

B és negatiu i C és positiu. No es pot produir sobreeiximent:

Suma $B + C$	
1 1 1 1 1 1	
1 0 0 0 0 1 1 1 0 1 (Ca2)	-4 8 3 (10)
+ 0 1 0 1 0 1 1 0 1 1 (Ca2)	+ 3 4 7 (10)
1 1 0 1 1 1 1 0 0 0 (Ca2)	-1 3 6 (10)
$B + C = 1101111000 = -136_{(10)}$	

26. Si tenim la cadena de bits 10110101, feu les conversions següents:

a) Considerant que representa un nombre en C_{a2} , representeu el mateix nombre en signe i magnitud i 16 bits:

Si la cadena de bits 10110101 representa un nombre en Ca2, es tracta d'un nombre negatiu, ja que el primer bit és 1. Podem fer un canvi de signe per obtenir la magnitud positiva:

```

| 1 0 1 1 0 1 0 1 ← Valor numèric inicial
|
|
| 0 1 0 0 1 0 1 0 ← Complement bit a bit de l'expressió inicial
|
+ |
+ | 1 ← Sumem 1 al bit menys significatiu del format
+-----+
| 0 1 0 0 1 0 1 1
|

```

Per codificar el valor inicial en signe i magnitud, tan sols hem de canviar el bit de signe del valor aconseguit amb el canvi de signe. Per tant, la codificació en signe i magnitud del valor 10110101 que està en Ca2 és **11001011**_{SM2}.

La codificació en 16 bits la podem obtenir fent l'extensió del format, que en aquest cas s'aconsegueix afegint els zeros necessaris a la dreta del signe:

$$11001011_{(\text{SM}2)} = 1000000001001011_{(\text{SM}2)}$$

b) Considerant que representa un nombre en signe i magnitud, representeu el mateix nombre en Ca2 i 16 bits:

Si es tracta d'un nombre codificat en signe i magnitud, és un nombre negatiu, ja que el primer dígit correspon al signe i és 1. La resta de dígits codifiquen la magnitud en binari. Podem obtenir la magnitud positiva si canviem el bit de signe: 00110101₂.

La representació dels valors positius coincideix en Ca2 i en signe i magnitud. Per tant, podem interpretar que tenim la magnitud positiva en Ca2 i que podem aconseguir la magnitud negativa en Ca2 aplicant un canvi de signe:

0	0	1	1	0	1	0	1	← Valor numèric inicial
1	1	0	0	1	0	1	0	← Complement bit a bit de l'expressió inicial
+							1	← Sumem 1 al bit menys significatiu del format
1	1	0	0	1	0	1	1	

Per tant, la codificació en Ca2 del valor 10110101_(SM2) que està en signe i magnitud és 11001011_(Ca2).

La codificació en 16 bits la podem obtenir si fem l'extensió del format, que en aquest cas s'aconsegueix copiant el bit de signe a l'esquerra de la codificació tantes vegades com sigui necessari:

$$11001011_{(Ca2)} = 1111111111001011_{(Ca2)}$$

27. Determineu quin valor decimal codifica la cadena de bits 1010010 en els supòsits següents:

a) Si es tracta d'un nombre en coma fixa sense signe de 7 bits dels quals 4 són fraccionaris

Amb aquest format, el nombre binari que representa aquesta tira de bits és $101,0010_2$ i, en aplicar el TFN, obtenim el nombre decimal que representa:

$$\begin{aligned} 101,0010_{(2)} &= 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} \\ &= 2^2 + 2^0 + 2^{-3} \\ &= 4 + 1 + 0,125 \\ &= 5,125_{(10)} \end{aligned}$$

b) Si es tracta d'un nombre en coma fixa sense signe de 7 bits dels quals 1 és fraccionari

En aquest cas, el nombre binari que representa aquesta tira de bits és $101001,0_{(2)}$ i, en aplicar el TFN, obtenim el nombre decimal que representa:

$$\begin{aligned} 101001,0_{(2)} &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} \\ &= 2^5 + 2^3 + 2^0 \\ &= 32 + 8 + 1 \\ &= 41_{(10)} \end{aligned}$$

28. Codifiqueu els nombres $+12,85_{(10)}$, $+0,7578125_{(10)}$ i $-11,025_{(10)}$ en una representació fraccionària binària en signe i magnitud de 8 bits dels quals 3 són fraccionaris. Feu servir una aproximació per arrodoniment en cas que sigui necessari.

Codifiquem el nombre $+12,85_{(10)}$ en el format especificat. Com que es tracta d'un nombre positiu, el bit de signe és 0. Pel que fa a la magnitud $12,85_{(10)}$, primer passem a binari la part entera i, posteriorment, la part fraccionària. Per a la part entera fem servir l'algorisme de divisions successives per la base d'arribada (2):

$$\begin{array}{rclcl} 12 & = & 6 \cdot 2 & + & 0 \\ 6 & = & 3 \cdot 2 & + & 0 \\ 3 & = & 1 \cdot 2 & + & 1 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \begin{array}{c} \uparrow \\ \\ \\ \end{array}$$

Així doncs, $12_{(10)} = 1100_{(2)}$. Pel que fa a la part fraccionària, fem multiplicacions successives per la base d'arribada (2):

$$\begin{array}{rclcl} 0,85 \cdot 2 & = & 1,70 & = & 1 + 0,70 \\ 0,70 \cdot 2 & = & 1,40 & = & 1 + 0,40 \\ 0,40 \cdot 2 & = & 0,80 & = & 0 + 0,80 \\ 0,80 \cdot 2 & = & 1,60 & = & 1 + 0,60 \\ 0,60 \cdot 2 & = & 1,20 & = & 1 + 0,20 \end{array} \quad \begin{array}{c} \\ \\ \\ \downarrow \end{array}$$

Com que el format especificat només té 3 bits per a la part fraccionària, ja tenim més bits dels necessaris i podem aturar el procés aquí. Per tant, $0,85_{(10)} = 0,11011..._{(2)}$. Per tal d'aproximar el valor amb 3 bits i arrodoniment, procedim de la manera següent:

$$0,11011_{(2)} + 0,0001_{(2)} = 0,11101_{(2)} \rightarrow \text{trunquem 3 bits} \rightarrow 0,111_{(2)}$$

Tot seguit, ajuntem les parts entera i fraccionària i obtenim: $12,85_{(10)} \approx 1100,111_{(2)}$. Finalment, per a obtenir la representació en el format indicat, cal afegir el bit de signe, de manera que la tira de bits que representa aquest nombre en el format donat és: **01100,111**₍₂₎. Cal recordar que la tira de bits que s'emmagatzemaria en un computador no conté la coma ni l'especificació de la base: **01100111**.

En els altres dos casos, procedim de manera totalment anàloga:

$+0,7578125_{(10)}$:

- El bit de signe és 0, perquè el nombre és positiu.
- La part entera és $0_{(10)} = 0_{(2)}$. Com que tenim 4 bits per a la part entera, escriurem 0000₍₂₎.
- Pel que fa a la part fraccionària:

$$\begin{array}{rclcl} 0,7578125 \cdot 2 & = & 1,515625 & = & 1 + 0,515625 \\ 0,515625 \cdot 2 & = & 1,03125 & = & 1 + 0,03125 \\ 0,03125 \cdot 2 & = & 0,0625 & = & 0 + 0,0625 \\ 0,0625 \cdot 2 & = & 0,125 & = & 0 + 0,125 \\ 0,1250 \cdot 2 & = & 0,25 & = & 0 + 0,25 \end{array} \quad \begin{array}{c} \\ \\ \\ \downarrow \end{array}$$

Així, $0,7578125_{(10)} = 0,11000..._{(2)}$ que, arrodonint amb 3 bits:

$$0,11000_{(2)} + 0,0001_{(2)} = 0,11010_{(2)} \rightarrow \text{trunquem 3 bits} \rightarrow 0,110_{(2)}$$

Finalment, ajuntem totes les parts i obtenim $0,7578125_{(10)} \approx 00000,110_{(2)}$. Per tant, la tira de bits que s'emmagatzemaria en un computador és **00000110**.

$-11,025_{(10)}$:

- El bit de signe és 1, perquè el nombre és negatiu.

- La part entera és $11_{(10)}$:

$$\begin{array}{rcll} 11 & = & 5 \cdot 2 & + 1 \\ 5 & = & 2 \cdot 2 & + 1 \\ 2 & = & 1 \cdot 2 & + 0 \\ 1 & = & 0 \cdot 2 & + 1 \end{array} \quad \begin{array}{c} \uparrow \\ \\ \\ \end{array}$$

És a dir $11_{(10)} = 1011_{(2)}$.

- Pel que fa a la part fraccionària:

$$\begin{array}{rcll} 0,025 \cdot 2 & = & 0,05 & = 0 + 0,05 \\ 0,05 \cdot 2 & = & 0,1 & = 0 + 0,1 \\ 0,1 \cdot 2 & = & 0,2 & = 0 + 0,2 \\ 0,2 \cdot 2 & = & 0,4 & = 0 + 0,4 \\ 0,4 \cdot 2 & = & 0,8 & = 0 + 0,8 \end{array} \quad \begin{array}{c} \\ \\ \\ \downarrow \end{array}$$

Així, $0,025_{(10)} = 0,00000..._{(2)}$ que, arrodonint amb 3 bits:

$$0,00000_{(2)} + 0,0001_{(2)} = 0,00010_{(2)} \rightarrow \text{trunquem 3 bits} \rightarrow 0,000_{(2)}$$

Finalment, ajuntem totes les parts i obtenim $-11,025_{(10)} \approx 11011,000_{(2)}$. Per tant, la tira de bits que s'emmagatzemaria en un computador és **11011000**.

29. Si tenim una representació en coma fixa binària en signe i magnitud de 8 bits dels quals 3 són fraccionaris, determineu els nombres codificats per les cadenes de bits 01001111, 11001111, 01010100, 00000000 i 10000000.

Per a obtenir la representació decimal d'aquestes cadenes de bits, només cal aplicar el TFN considerant que els tres darrers bits constitueixen la part fraccionària i que el primer bit representa el signe:

$$01001111 \rightarrow +1001,111_{(2)}$$

$$\begin{aligned} +1001,111_{(2)} &= 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} \\ &= 2^3 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3} \\ &= 8 + 1 + 0,5 + 0,25 + 0,125 \\ &= +9,875_{(10)} \end{aligned}$$

$$11001111 \rightarrow -1001,111_{(2)}$$

$$\begin{aligned} -1001,111_{(2)} &= -(1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3}) \\ &= -(2^3 + 2^0 + 2^{-1} + 2^{-2} + 2^{-3}) \\ &= -(8 + 1 + 0,5 + 0,25 + 0,125) \\ &= -9,875_{(10)} \end{aligned}$$

$$01010100 \rightarrow +1010,100_{(2)}$$

$$\begin{aligned} +1010,100_{(2)} &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} \\ &= 2^3 + 2^1 + 2^{-1} \\ &= 8 + 2 + 0,5 \\ &= +10,5_{(10)} \end{aligned}$$

$$00000000 \rightarrow +0000,000_{(2)}$$

$$\begin{aligned} +0000,000_{(2)} &= 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} \\ &= +0_{(10)} \end{aligned}$$

$$10000000 \rightarrow -0000,000_{(2)}$$

$$\begin{aligned} -0000,000_{(2)} &= -(0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3}) \\ &= -0_{(10)} \end{aligned}$$

30. Si les cadenes de bits 00101010, 11010010 i 10100010 representen nombres en coma fixa sense signe de 8 bits dels quals 3 són fraccionaris, representeu-los en un format de coma fixa sense signe de 12 bits dels quals 4 són fraccionaris.

Per estendre el rang d'aquestes tires de bits, cal afegir zeros tant a l'esquerra com a la dreta de la magnitud, atès que es tracta de magnituds sense signe.

La representació original té 5 bits per a la part entera i la representació d'arribada en té 8. Per tant, cal afegir $8 - 5 = 3$ bits a l'esquerra de la magnitud. De la mateixa manera, la represen-

tació original té 3 bits per a la part fraccionària, mentre que la representació d'arribada en té 4. Així doncs, cal afegir $4 - 3 = 1$ bit a la dreta de la magnitud:

00101010 \rightarrow 00101,010 \rightarrow 00000101,0100 \rightarrow 000001010100
 11010010 \rightarrow 11010,010 \rightarrow 00011010,0100 \rightarrow 000110100100
 10100010 \rightarrow 10100,010 \rightarrow 00010100,0100 \rightarrow 000101000100

31. Repetiu l'activitat anterior considerant que es tracta de nombres en signe i magnitud.

En aquest cas, cal considerar que el primer bit representa el signe. L'extensió de la part entera implica replicar el bit de signe i inserir tants zeros com calgui. En aquest cas passem de 4 a 7 bits per a la part entera, per tant, caldrà afegir $7 - 4 = 3$ bits a l'esquerra de la part entera. Pel que fa a la part fraccionària, la situació és idèntica a l'exercici anterior. Així doncs, caldrà afegir $4 - 3 = 1$ bit a la dreta de la magnitud:

00101010 \rightarrow +0101,010 \rightarrow +0000101,0100 \rightarrow 000001010100
 11010010 \rightarrow -1010,010 \rightarrow -0001010,0100 \rightarrow 100010100100
 10100010 \rightarrow -0100,010 \rightarrow -0000100,0100 \rightarrow 100001000100

32. Determineu el rang de representació i la precisió en els formats següents:

a) Coma fixa en signe i magnitud amb 8 bits dels quals 3 són fraccionaris.

El rang d'una representació fraccionària amb signe i magnitud és:

$$[-2^{n-m-1} + 2^{-m}, +2^{n-m-1} - 2^{-m}]$$

en el qual n és el nombre de bits emprat en la representació i m el nombre dígits fraccionaris. Per tant, el rang en aquest cas l'obtenim fent $n = 8$ i $m = 3$, és a dir:

$$[-2^{8-3-1} + 2^{-3}, +2^{8-3-1} - 2^{-3}] = [-2^4 + 2^{-3}, +2^4 - 2^{-3}] = [-15,875_{(10)}, +15,875_{(10)}].$$

Pel que fa a la precisió, la determina el bit de menys pes de la magnitud i, concretament, és igual a $2^{-m} = 2^{-3} = 0,125_{(10)}$.

b) Coma fixa en signe i magnitud amb 8 bits dels quals 4 són fraccionaris.

En aquest cas, la situació és la mateixa que en l'apartat anterior però amb $n = 8$ i $m = 4$. El rang és el següent:

$$[-2^{8-4-1} + 2^{-4}, +2^{8-4-1} - 2^{-4}] = [-2^3 + 2^{-4}, +2^3 - 2^{-4}] = [-7,9375_{(10)}, +7,9375_{(10)}].$$

Anàlogament, la precisió és igual a $2^{-m} = 2^{-4} = 0,0625_{(10)}$.

c) Coma fixa sense signe amb 8 bits dels quals 3 són fraccionaris.

En coma fixa sense signe, el rang de representació es determina per:

$$[0, +2^{n-m} - 2^{-m}]$$

en el qual n és el nombre de bits emprat en la representació i m el nombre de dígits fraccionaris. Per tant, el rang en aquest cas l'obtenim fent $n = 8$ i $m = 3$, és a dir:

$$[0, +2^{8-3} - 2^{-3}] = [0, +2^5 - 2^{-3}] = [0, +31,875_{(10)}].$$

La precisió és la mateixa que en els casos anteriors, perquè és independent de si la representació és sense signe o en signe i magnitud. Per tant, la precisió és $2^{-m} = 2^{-3} = 0,125$.

d) Coma fixa sense signe amb 8 bits dels quals 4 són fraccionaris.

En aquest cas, la situació és la mateixa que en l'apartat anterior però amb $n = 8$ i $m = 4$. El rang és el següent:

$$[0, +2^{8-4} - 2^{-4}] = [0, +2^4 - 2^{-4}] = [0, +15,9375_{(10)}].$$

Anàlogament, la precisió és $2^{-m} = 2^{-4} = 0,0625_{(10)}$.

33. Determineu la precisió necessària per poder representar el nombre $+0,1875_{(10)}$ de manera exacta (sense error de representació) amb un format de coma fixa en base 2.

Per tal de determinar la precisió necessària per a representar el nombre $+0,1875_{(10)}$ de manera exacta, primer codificarem en binari aquest nombre fent multiplicacions successives per la base d'arribada:

$$\begin{array}{rclcl} 0,1875 \cdot 2 & = & 0,375 & = & 0 + 0,375 \\ 0,375 \cdot 2 & = & 0,75 & = & 0 + 0,75 \\ 0,75 \cdot 2 & = & 1,5 & = & 1 + 0,5 \\ 0,5 \cdot 2 & = & 1 & = & 1 + 0 \end{array} \quad \downarrow$$

Per tant, $0,1875_{(10)} = 0,0011_{(2)}$. Així doncs, per tal de poder representar aquest nombre de manera exacta, és necessari que la representació disposi, com a mínim, de 4 dígits fraccionaris ($m = 4$) i, per tant, una precisió de la forma $2^{-m} = 2^{-4} = 0,0625_{(10)}$.

34. Determineu les característiques de rang i precisió, i també el nombre de dígits enters i fraccionaris necessaris en un format de coma fixa en signe i magnitud, per poder representar de manera exacta els nombres $+31,875_{(10)}$ i $-16,21875_{(10)}$.

En primer lloc, obtindrem la codificació binària d'aquests dos nombres.

Per al $+31,875_{(10)}$:

- El bit de signe és 0, perquè el nombre és positiu.
- La part entera és $31_{(10)}$:

$$\begin{array}{rclcl} 31 & = & 15 \cdot 2 & + & 1 \\ 15 & = & 7 \cdot 2 & + & 1 \\ 7 & = & 3 \cdot 2 & + & 1 \\ 3 & = & 1 \cdot 2 & + & 1 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \uparrow$$

És a dir $31_{(10)} = 11111_{(2)}$.

- Pel que fa a la part fraccionària:

$$\begin{array}{rclcl} 0,875 \cdot 2 & = & 1,75 & = & 1 + 0,75 \\ 0,75 \cdot 2 & = & 1,5 & = & 1 + 0,5 \\ 0,5 \cdot 2 & = & 1 & = & 1 + 0 \end{array} \quad \downarrow$$

Així doncs, $0,875_{(10)} = 0,111_{(2)}$

- Finalment, ajuntem totes les parts i obtenim $+31,875_{(10)} = 011111,111_{(SM2)}$.

Per al $-16,21875_{(10)}$:

- El bit de signe és 1, perquè el nombre és negatiu.
- La part entera és $16_{(10)}$:

$$\begin{array}{rclcl} 16 & = & 8 \cdot 2 & + & 0 \\ 8 & = & 4 \cdot 2 & + & 0 \\ 4 & = & 2 \cdot 2 & + & 0 \\ 2 & = & 1 \cdot 2 & + & 0 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \uparrow$$

És a dir $16_{(10)} = 10000_{(2)}$.

- Pel que fa a la part fraccionària:

$$\begin{array}{rclcl} 0,21875 \cdot 2 & = & 0,4375 & = & 0 + 0,4375 \\ 0,4375 \cdot 2 & = & 0,875 & = & 0 + 0,875 \\ 0,875 \cdot 2 & = & 1,75 & = & 1 + 0,75 \\ 0,75 \cdot 2 & = & 1,5 & = & 1 + 0,5 \\ 0,5 \cdot 2 & = & 1 & = & 1 + 0 \end{array} \quad \downarrow$$

Així doncs, $0,21875_{(10)} = 0,00111_{(2)}$

- Finalment, ajuntem totes les parts i obtenim $-16,21875_{(10)} = 110000,00111_{(SM2)}$

Per a representar exactament **tots dos** nombres, cal un format que tingui:

- Un bit de signe

- 5 bits per a la part entera (amb 5 bits es poden representar exactament tant $16_{(10)}$ com $31_{(10)}$).
- 5 bits per a la part fraccionària (amb 5 bits es poden representar exactament tant $0,875_{(10)}$ com $0,21875_{(10)}$).

Tenim, doncs, un format de signe i magnitud amb $n = 11$ (bit de signe, 5 bits de part entera i 5 bits de part fraccionària) i $m = 5$. Amb aquestes dades, el rang és el següent:
 $[-2^{n-m-1} + 2^{-m}, +2^{n-m-1} - 2^{-m}] = [-2^{11-5-1} + 2^{-5}, +2^{11-5-1} - 2^{-5}] = [-2^5 + 2^{-5}, +2^5 - 2^{-5}] =$
 $= [-31,96875_{(10)}, +31,96875_{(10)}]$. La precisió és $2^{-m} = 2^{-5} = 0,03125_{(10)}$.

35. Feu la suma i la resta dels parells de nombres següents assumint que estan en coma fixa en signe i magnitud amb 8 bits dels quals 3 són fraccionaris. Verifiqueu si el resultat és correcte:

a) 00111000_2 i 10100000_2

$$00111000_2 \rightarrow +0111,000_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 4 + 2 + 1 = +7_{(10)}$$

$$10100000_2 \rightarrow -0100,000_2 = -1 \cdot 2^2 = -4_{(10)}$$

Per calcular la suma d'aquests dos nombres cal, en primer lloc, adonar-nos que es tracta de nombres de diferents signe. Per tant, la suma l'obtidrem si restem la magnitud més gran ($0111,000_2$) de la més petita ($0100,000_2$) i si copiem el signe de la magnitud més gran (positiu):

$$\begin{array}{r}
 0 \ 1 \ 1 \ 1 \ , \ 0 \ 0 \ 0 \ 0_2 \\
 0 \ 0 \ 0 \ 0 \ , \ 0 \ 0 \ 0 \ 0 \quad \leftarrow \text{ròssecs} \\
 - \quad 0 \ 1 \ 0 \ 0 \ , \ 0 \ 0 \ 0 \ 0_2 \\
 \hline
 0 \ 0 \ 1 \ 1 \ , \ 0 \ 0 \ 0 \ 0_2 \quad \leftarrow \text{resultat}
 \end{array}$$

Per tant, $00111000_2 + 10100000_2 \rightarrow +0111,000_2 + (-0100,000_2) = +0011,000_2 \rightarrow 00011000_2$. Si convertim el resultat a base 10 tenim que $+0011,000_2 = 1 \cdot 2^1 + 1 \cdot 2^0 = 2 + 1 = 3_{(10)}$, cosa que demostra que el resultat és correcte, ja que $+7_{(10)} + (-4_{(10)}) = 3_{(10)}$.

Per calcular la resta caldrà fer la suma de les magnituds, perquè el nombre 10100000_2 és negatiu. El resultat de la resta serà positiu (perquè a un nombre positiu li'n restem un de negatiu):

$$\begin{array}{r}
 1 \ 0 \ 0 \ 0 \ , \ 0 \ 0 \ 0 \ 0 \quad \leftarrow \text{ròssecs} \\
 0 \ 1 \ 1 \ 1 \ , \ 0 \ 0 \ 0 \ 0_2 \\
 + \quad 0 \ 1 \ 0 \ 0 \ , \ 0 \ 0 \ 0 \ 0_2 \\
 \hline
 1 \ 0 \ 1 \ 1 \ , \ 0 \ 0 \ 0 \ 0_2 \quad \leftarrow \text{resultat}
 \end{array}$$

Per tant, $00111000_2 - 10100000_2 \rightarrow +0111,000_2 - (-0100,000_2) = +1011,000_2 \rightarrow 01011000_2$. Novament, podem comprovar que el resultat obtingut és correcte si el convertim a base 10: $+1011,000_2 = 2^3 + 2^1 + 2^0 = 8 + 2 + 1 = +11_{(10)} = +7_{(10)} - (-4_{(10)})$.

b) 10111010_2 i 11101100_2

$$10111010_2 \rightarrow -0111,010_2 = -(2^2 + 2^1 + 2^0 + 2^{-2}) = -(4 + 2 + 1 + 0,25) = -7,25_{(10)}$$

$$11101100_2 \rightarrow -1101,100_2 = -(2^3 + 2^2 + 2^0 + 2^{-1}) = -(8 + 4 + 1 + 0,5) = -13,5_{(10)}$$

En aquest cas es tracta de dos nombres negatius, atès que el primer dígit, que indica el signe, és un 1 en tots dos casos. Per tal de fer la suma, caldrà fer la suma de les magnituds i copiar el bit signe:

$$\begin{array}{r}
 1 \ 1 \ 1 \ 1 \ 0 \ , \ 0 \ 0 \ 0 \ 0 \quad \leftarrow \text{ròssecs} \\
 0 \ 1 \ 1 \ 1 \ , \ 0 \ 1 \ 0 \ 0_2 \\
 + \quad 1 \ 1 \ 0 \ 1 \ , \ 1 \ 0 \ 0 \ 0_2 \\
 \hline
 1 \ 0 \ 1 \ 0 \ 0 \ , \ 1 \ 1 \ 0 \ 0_2 \quad \leftarrow \text{resultat}
 \end{array}$$

Observeu que s'ha produït ròssec en l'últim bit i, per tant, com que estem sumant només les magnituds, hi ha sobreiximent. Això vol dir que el resultat de la suma no és representable amb 8 bits dels quals 3 són fraccionaris.

Si disposéssim d'un bit més per a la part entera (un format de 9 bits amb 3 d'ells fraccionaris) el resultat sí que seria representable: tindríem $-10100,110_2 \rightarrow 110100110$. Efectivament comprovem que

$$-10100,110_2 = -(2^4 + 2^2 + 2^{-1} + 2^{-2}) = -(16 + 4 + 0,5 + 0,25) = -20,75_{(10)} = -7,25_{(10)} + (-13,5_{(10)}).$$

Pel que fa a la resta, cal adonar-se que el subtrahend ($-1101,100_2$) és més gran en magnitud que el minuend ($-0111,010_2$). Així doncs, farem la resta de les magnituds com $1101,100_2 - 0111,010_2$ i el resultat ha de ser positiu:

$$\begin{array}{r} 1\ 1\ 0\ 1\ ,\ 1\ 0\ 0\ _2 \\ 1\ 1\ 0\ 0\ \quad 1\ 0\ \quad \leftarrow \text{ròssecs} \\ -\quad 0\ 1\ 1\ 1\ ,\ 0\ 1\ 0\ _2 \\ \hline 0\ 1\ 1\ 0\ ,\ 0\ 1\ 0\ _2 \leftarrow \text{resultat} \end{array}$$

Per tant, $10111,010_2 - 11101100_2 \rightarrow -0111,010_2 - (-1101100_2) = +0110,010_2 \rightarrow 0110010_2$. Novament, podem comprovar que el resultat obtingut és correcte si el convertim a base 10:

$$+0110,010_2 = 2^2 + 2^1 + 2^{-2} = 4 + 2 + 0,25 = +6,25_{(10)} = -7,25_{(10)} - (-13,5_{(10)}) = -7,25_{(10)} + 13,5_{(10)}.$$

36. Codifiqueu en BCD el nombre $125_{(10)}$.

Primerament codifiquem en binari amb 4 bits cadascun del dígit del nombre:

$$\begin{aligned} 1_{(10)} &= 0001_2 \\ 2_{(10)} &= 0010_2 \\ 5_{(10)} &= 0101_2 \end{aligned}$$

I, finalment, formem una única tira amb tots els bits **000100100101**₂.

37. Codifiqueu en BCD el nombre $637_{(10)}$.

Com en l'apartat anterior, primerament codifiquem en binari amb quatre bits cadascun del dígit del nombre:

$$\begin{aligned} 6_{(10)} &= 0110_2 \\ 3_{(10)} &= 0011_2 \\ 7_{(10)} &= 0111_2 \end{aligned}$$

I, finalment, formem una única tira amb tots els bits **011000110111**₂.

38. Indiqueu quin nombre codifica la representació BCD següent: 000100111000.

Separem la tira de bits en grups de 4 i comprovem quin nombre decimal representen. Tenim $000100111000_2 \rightarrow 0001 \mid 0011 \mid 1000$, per tant:

$$\begin{aligned} 0001_2 &= 1_{(10)} \\ 0011_2 &= 3_{(10)} \\ 1000_2 &= 8_{(10)} \end{aligned}$$

I, finalment, construïm el nombre decimal ajuntant els dígit decimal: **138**₍₁₀₎.

39. Codifiqueu el nombre $427_{(10)}$ en BCD i en binari. Compareu el nombre de bits necessari en els dos casos.

Per codificar $427_{(10)}$ en BCD, primerament codifiquem en binari amb 4 bits cadascun del dígit del nombre:

$$\begin{aligned} 4_{(10)} &= 0100_2 \\ 2_{(10)} &= 0010_2 \\ 7_{(10)} &= 0111_2 \end{aligned}$$

I, finalment, formem una única tira amb tots els bits **010000100111**₂.

Per a codificar el mateix nombre en binari cal aplicar el mètode de les divisions successives per la base d'arribada (2):

$$\begin{array}{rclcl}
 427 & = & 213 & \cdot & 2 & + & 1 \\
 213 & = & 106 & \cdot & 2 & + & 1 \\
 106 & = & 53 & \cdot & 2 & + & 0 \\
 53 & = & 26 & \cdot & 2 & + & 1 \\
 26 & = & 13 & \cdot & 2 & + & 0 \\
 13 & = & 6 & \cdot & 2 & + & 1 \\
 6 & = & 3 & \cdot & 2 & + & 0 \\
 3 & = & 1 & \cdot & 2 & + & 1 \\
 1 & = & 0 & \cdot & 2 & + & 1
 \end{array}
 \quad \uparrow$$

Per tant, $427_{(10)} = 110101011_{(2)}$.

En el cas de la codificació BCD calen **12 bits** per a representar el nombre $427_{(10)}$, en canvi, la codificació binària només en necessita 9.

40. Trobar el valor decimal que codifiquen les cadenes de bits següents, interpretant que es tracta de nombres en un format de coma flotant de 8 bits amb mantissa normalitzada de la forma $1,X$ i amb bit implícit:

a) 11110010, en el qual la mantissa és de 4 bits.

En primer lloc, cal identificar els diferents elements que componen aquest format:

Signe (S)	Exponent (e)	Mantissa
1	111	0010

- Bit de signe $s = 1$, per tant, es tracta d'un nombre negatiu.
- Exponent: com que la mantissa és de 4 bits, queden 3 bits ($111_{(2)}$) per a l'exponent ($q = 3$). Per tant, l'excés serà $M = 2^{q-1} = 2^2 = 4_{(10)}$. Llavors, l'exponent val $e = 111_{(2)} - 4_{(10)} = (2^2 + 2^1 + 2^0) - 4 = (4 + 2 + 1) - 4 = 7 - 4 = 3_{(10)}$.
- Finalment, els quatre bits que queden formen la mantissa que, com que està normalitzada en la forma $1,X$ amb bit implícit, pren el valor $R = 1,0010_{(2)}$.

Ara ja podem calcular el nombre representat en base 10 aplicant la fórmula $\pm R \cdot 2^e$ i el TFN:

$$-1,0010_{(2)} \cdot 2^3 = -(2^0 + 2^{-3}) \cdot 2^3 = -(2^3 + 2^0) = -(8 + 1) = -9_{(10)}.$$

b) 01010011, on la mantissa és de 3 bits.

En aquest cas, procedim de manera totalment anàloga a l'apartat anterior:

Signe (S)	Exponent (e)	Mantissa
0	1010	011

- Bit de signe $s = 0$, per tant, es tracta d'un nombre positiu.
- Exponent: com que la mantissa és de 3 bits, queden 4 bits ($1010_{(2)}$) per a l'exponent ($q = 4$). Per tant, l'excés serà $M = 2^{q-1} = 2^3 = 8_{(10)}$. Llavors, l'exponent val $e = 1010_{(2)} - 8_{(10)} = (2^3 + 2^1) - 8 = (8 + 2) - 8 = 10 - 8 = 2_{(10)}$.
- Finalment, els quatre bits que queden formen la mantissa que, com que està normalitzada en la forma $1,X$ amb bit implícit, pren el valor $R = 1,011_{(2)}$.

Ara ja podem calcular el nombre representat en base 10 aplicant la fórmula $\pm R \cdot 2^e$ i el TFN:

$$+1,011_{(2)} \cdot 2^2 = +(2^0 + 2^{-2} + 2^{-3}) \cdot 2^2 = +(2^2 + 2^0 + 2^{-1}) = +(4 + 1 + 0,5) = +5,5_{(10)}.$$

41. Feu les codificacions següents:

a) El nombre $-1,335_{(10)}$ en coma flotant de 8 bits, mantissa de 3 bits normalitzada de la forma $1,X$ i amb bit implícit emprant una aproximació per truncament.

El format especificat té la forma donada per la taula següent:

Signe (S)	Exponent (e)	Mantissa
1 bit	4 bits	3 bits

En primer lloc, codificarem la magnitud $1,335_{(10)}$ en base 2:

- La part entera és $1_{(10)}$:

$$1 = 0 \cdot 2 + 1$$

És a dir $1_{(10)} = 1_{(2)}$.

- Pel que fa a la part fraccionària:

$$\begin{array}{rclcl} 0,335 & \cdot & 2 & = & 0,67 & = & 0 + 0,67 \\ 0,67 & \cdot & 2 & = & 1,34 & = & 1 + 0,34 \\ 0,34 & \cdot & 2 & = & 0,68 & = & 0 + 0,68 \\ 0,68 & \cdot & 2 & = & 1,36 & = & 1 + 0,36 \\ 0,36 & \cdot & 2 & = & 0,72 & = & 0 + 0,72 \\ 0,72 & \cdot & 2 & = & 1,44 & = & 1 + 0,44 \end{array} \quad \downarrow$$

Així doncs, $0,335_{(10)} = 0,010101..._{(2)}$ i ens aturem perquè ja tenim més bits dels que permet representar aquest format.

- Ara, ajuntem les dues parts de la magnitud i obtenim $1,335_{(10)} \approx 1,010101_{(2)}$.
- Seguidament normalitzem l'expressió obtinguda (afegint la informació del signe) en la forma $\pm R \cdot 2^e$: $+1,335_{(10)} \approx +1,010101_{(2)} \cdot 2^0$.
- Finalment, obtenim els diferents elements que defineixen la representació:
 - Signe:** es tracta d'un nombre negatiu, per tant, el bit de signe serà un 1.
 - Exponent:** l'exponent és $0_{(10)}$ que cal codificar en excés a $M = 2^q - 1$. Com que disposem de 4 bits, l'excés és $M = 2^3 = 8$. Per tant, cal representar $0_{(10)} + 8_{(10)} = 8_{(10)}$ amb 4 bits:

$$\begin{array}{rclcl} 8 & = & 4 \cdot 2 & + & 0 \\ 4 & = & 2 \cdot 2 & + & 0 \\ 2 & = & 1 \cdot 2 & + & 0 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \uparrow$$

És a dir $e = 8_{(10)} = 1000_{(2)}$ (en excés a 8).

- Mantissa:** $R = 1,010101_{(2)}$. Com que hi ha bit implícit (1,X), només cal representar la part de la dreta de la coma i disposem de 3 bits (amb truncament). Així doncs, la mantissa seran els tres bits a partir de la coma: **010**.
- Ara ja podem ajuntar totes les parts:

Signe (S)	Exponent (e)	Mantissa
1	1000	010

És a dir, la representació és **11000010**.

b) Repetiu l'apartat anterior, però amb una aproximació per arrodoniment.

L'única diferència amb l'apartat anterior és el càlcul de la mantissa, perquè ara cal arrodonir en comptes de truncar. Tal com s'indica més amunt, tenim $R = 1,010101_{(2)}$; per arrodonir-ho amb tres bits, cal sumar-hi la meitat de la precisió i truncar el resultat:

$$1,010101_{(2)} + 0,0001_{(2)} = 1,011001_{(2)} \rightarrow \text{trunquem 3 bits} \rightarrow 1,011_{(2)}$$

Per tant, la representació serà la següent:

Signe (S)	Exponent (e)	Mantissa
1	1000	011

És a dir, la seqüència de bits de la representació és **11000011**.

c) El nombre $10,0327_{(10)}$ en coma flotant de 9 bits, mantissa de 3 bits normalitzada de la forma $1,X$ i amb bit implícit emprant una aproximació per truncament.

El format especificat té la forma donada per la taula següent:

Signe (<i>S</i>)	Exponent (<i>e</i>)	Mantissa
1 bit	5 bits	3 bits

En primer lloc, codificarem la magnitud $10,0327_{(10)}$ en base 2:

- La part entera és $10_{(10)}$:

$$\begin{array}{rclcl} 10 & = & 5 \cdot 2 & + & 0 \\ 5 & = & 2 \cdot 2 & + & 1 \\ 2 & = & 1 \cdot 2 & + & 0 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \begin{array}{c} \uparrow \\ \\ \\ \end{array}$$

És a dir $10_{(10)} = 1010_{(2)}$.

- Pel que fa a la part fraccionària:

$$\begin{array}{rclcl} 0,0327 & \cdot & 2 & = & 0,0654 = 0 + 0,0654 \\ 0,0654 & \cdot & 2 & = & 0,1308 = 0 + 0,1308 \\ 0,1308 & \cdot & 2 & = & 0,2616 = 0 + 0,2616 \\ 0,2616 & \cdot & 2 & = & 0,5232 = 0 + 0,5232 \\ 0,5232 & \cdot & 2 & = & 1,0464 = 1 + 0,0464 \\ 0,0464 & \cdot & 2 & = & 0,0928 = 0 + 0,0928 \end{array} \quad \begin{array}{c} \\ \\ \\ \\ \downarrow \end{array}$$

Així doncs, $0,0327_{(10)} = 0,000010..._{(2)}$ i ens aturem perquè ja tenim més bits dels que permet representar aquest format.

- Ara, ajuntem les dues parts de la magnitud i obtenim $10,0327_{(10)} \approx 1010,000010_{(2)}$.
- Seguidament normalitzem l'expressió obtinguda (afegint la informació del signe) en la forma $\pm R \cdot 2^e$: $+10,0327_{(10)} \approx +1,01000010_{(2)} \cdot 2^3$.
- Finalment, obtenim els diferents elements que defineixen la representació:
 - Signe:** es tracta d'un nombre positiu, per tant, el bit de signe serà un 0.
 - Exponent:** l'exponent és $3_{(10)}$ que cal codificar en excés a $M = 2^q - 1$. Com que disposem de 5 bits, l'excés és $M = 2^4 = 16$. Per tant, cal representar $3_{(10)} + 16_{(10)} = 19_{(10)}$ amb 4 bits:

$$\begin{array}{rclcl} 19 & = & 9 \cdot 2 & + & 1 \\ 9 & = & 4 \cdot 2 & + & 1 \\ 4 & = & 2 \cdot 2 & + & 0 \\ 2 & = & 1 \cdot 2 & + & 0 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \begin{array}{c} \uparrow \\ \\ \\ \\ \end{array}$$

És a dir $e = 19_{(10)} = 10011_{(2)}$ (en excés a 16).

- Mantissa:** $R = 1,01000010_{(2)}$. Com que hi ha bit implícit (1,X), només cal representar la part de la dreta de la coma i disposem de 3 bits (amb truncament). Així doncs, la mantissa seran els tres bits a partir de la coma: **010**.
- Ara ja podem ajuntar totes les parts:

Signe (<i>S</i>)	Exponent (<i>e</i>)	Mantissa
0	10011	010

És a dir, la representació és **010011010**.

42. Determineu si el nombre $2,89_{(10)} \cdot 10^{10}$ és representable en un format de coma flotant de 16 bits, amb mantissa normalitzada de la forma $1,X$, bit implícit i 5 bits per a l'exponent.

El format especificat té la forma donada per la taula següent:

Signe (<i>s</i>)	Exponent (<i>e</i>)	Mantissa
1 bit	5 bits	10 bits

El nombre positiu de magnitud més gran que podem representar en aquest format té un 0 en el bit de signe i un 1 en la resta de bits: 0111111111111111.

Cerquem el nombre decimal que representa:

- Bit de signe $s = 0$, per tant, es tracta d'un nombre positiu.
- Exponent: 11111_2 , és a dir, $q = 5$. Per tant, l'excés serà $M = 2^{q-1} = 2^4 = 16_{(10)}$. Llavors, l'exponent val $e = 11111_2 - 16_{(10)} = 31 - 16 = 15_{(10)}$.
- Finalment, els bits que resten formen la mantissa que, com que està normalitzada en la forma $1,X$ amb bit implícit pren el valor $R = 1,111111111_2$.

Ara ja podem calcular el nombre representat en base 10 aplicant la fórmula $\pm R \cdot 2^e$ i el TFN: $+1,111111111_2 \cdot 2^{15} = +1111111111_2 \cdot 2^5 = +2047 \cdot 2^5 = +65504_{(10)}$.

Com que el nombre $2,89_{(10)} \cdot 10^{10}$ és més gran que el $65504_{(10)}$, no es pot representar en aquest format.

43. Determineu si el nombre $-1256_{(10)} \cdot 10^{-2}$ és representable en un format de coma flotant de 10 bits, amb mantissa normalitzada de la forma $1,X$, bit implícit i 6 bits per a l'exponent.

El format especificat té la forma donada per la taula següent:

Signe (s)	Exponent (e)	Mantissa
1 bit	6 bits	3 bits

El nombre negatiu de magnitud més gran que podem representar en aquest format té tots els bits a 1: 1111111111.

Cerquem el nombre decimal que representa:

- Bit de signe $s = 1$, per tant, es tracta d'un nombre negatiu.
- Exponent: 11111_2 , és a dir, $q = 6$. Per tant, l'excés serà $M = 2^{q-1} = 2^5 = 32_{(10)}$. Llavors, l'exponent val $e = 11111_2 - 32_{(10)} = 63 - 32 = 31_{(10)}$.
- Finalment, els bits que resten formen la mantissa que, com que està normalitzada en la forma $1,X$ amb bit implícit pren el valor $R = 1,111_2$.

Ara ja podem calcular el nombre representat en base 10 tot aplicant la fórmula $\pm R \cdot 2^e$ i el TFN:

$$-1,111_2 \cdot 2^{31} = -1111_2 \cdot 2^{28} = -15 \cdot 2^{28}_{(10)}.$$

Com que la magnitud del nombre $-1256_{(10)} \cdot 10^{-2}$ (o, el que és el mateix, del $-12,56_{(10)}$) és més petita que $-15 \cdot 2^{28}_{(10)}$, es pot representar en aquest format.

Exercicis d'autoavaluació

1. Podem trobar la codificació en Ca2 i 8 bits del nombre $-10_{(10)}$ de les maneres següents:

- $2^8 - 10 = 256_{(10)} - 10_{(10)} = 246_{(10)} = 11110110_{(Ca2)}$, o bé
- $2^8 - 10 = 100000000_2 - 1010_2 = 11110110_{(Ca2)}$, o bé
- $+10_{(10)} = +1010_2 \rightarrow$ representació de la magnitud positiva $\rightarrow 00001010_{(Ca2)} \rightarrow$
 \rightarrow canvi de signe $\rightarrow 11110110_{(Ca2)}$

Per a codificar el nombre en signe i magnitud, hem d'afegir a la representació de la magnitud en base 2 la informació del signe. Com que és un nombre negatiu, el bit de signe serà 1. Per tant, la codificació serà **10001010**_(SM2).

2.

a) Si es tracta d'un nombre codificat en complement a 2.

En aquest cas es tracta d'un nombre positiu, perquè el bit de l'extrem esquerre és 0. Per tant, la resta de bits codifiquen la magnitud com en el cas de signe i magnitud. Si apliquem el TFN a la magnitud obtenim:

$$0100100_2 = 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 36_{(10)}$$

En aquest cas codifica el nombre decimal +36.

b) Si es tracta d'un nombre codificat en signe i magnitud.

Com que es tracta d'un nombre positiu, i la codificació en signe i magnitud d'un nombre positiu coincideix amb la representació en Ca2, la cadena codifica el mateix nombre, el $+36_{(10)}$.

3.

$$\begin{array}{r}
 1 \ 1 \ 1 \\
 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \quad (2 \\
 + \quad 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \quad (2 \\
 \hline
 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \quad (2
 \end{array}$$

El que podem observar és que necessitem un bit més per a poder representar el resultat de l'operació.

4. Com que es tracta d'un nombre positiu, el bit de signe és 0. Pel que fa a la magnitud $12,346_{(10)}$, primer passem a binari la part entera i, posteriorment, la part fraccionària. Per a la part entera fem servir l'algorisme de divisions successives per la base d'arribada (2):

$$\begin{array}{rclcl}
 12 & = & 6 \cdot 2 & + & 0 \\
 6 & = & 3 \cdot 2 & + & 0 \\
 3 & = & 1 \cdot 2 & + & 1 \\
 1 & = & 0 \cdot 2 & + & 1
 \end{array}
 \begin{array}{c} \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \end{array}$$

Així doncs, $12_{(10)} = 1100_{(2)}$. Pel que fa a la part fraccionària, fem multiplicacions successives per la base d'arribada (2):

$$\begin{array}{rclcl}
 0,346 \cdot 2 & = & 0,692 & = & 0 + 0,692 \\
 0,692 \cdot 2 & = & 1,384 & = & 1 + 0,384 \\
 0,384 \cdot 2 & = & 0,768 & = & 0 + 0,768 \\
 0,768 \cdot 2 & = & 1,536 & = & 1 + 0,536 \\
 0,536 \cdot 2 & = & 1,072 & = & 1 + 0,072
 \end{array}
 \begin{array}{c} \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \\ \downarrow \end{array}$$

Com que el format especificat només té 3 bits per a la part fraccionària, ja tenim més bits dels necessaris i podem aturar el procés aquí. Per tant, $0,346_{(10)} = 0,01011..._{(2)}$. Per tal d'aproximar el valor amb 3 bits per truncament eliminem tots els bits a partir del tercer: $0,346_{(10)} = 0,010_{(2)}$.

Tot seguit, ajuntem les parts entera i fraccionària i obtenim $12,346_{(10)} \approx 1100,010_{(2)}$. Finalment, per a obtenir la representació en el format indicat, cal afegir el bit de signe, de manera que la tira de bits que representa aquest nombre en el format donat és $01100,010_{(2)}$. Cal recordar que la tira de bits que s'emmagatzemaria en un ordinador no conté la coma ni l'especificació de la base: **01100010**.

5. Podem obtenir la representació en Ca2 i 8 bits del $-45_{(10)}$ de qualsevol de les formes següents:

- $2^8 - 45 = 256_{(10)} - 45_{(10)} = 211_{(10)} = 11010011_{(Ca2)}$, o bé
- $2^8 - 45 = 100000000_{(2)} - 101101_{(2)} = 11010011_{(Ca2)}$, o bé
- $+45_{(10)} = +101101_{(2)} \rightarrow$ representació de la magnitud positiva $\rightarrow 00101101_{(Ca2)} \rightarrow$ canvi de signe $\rightarrow 11010011_{(Ca2)}$

6. En primer lloc, obtindrem la codificació binària del nombre $-35,25_{(10)}$:

- El bit de signe és 1, perquè el nombre és negatiu.
- La part entera és $35_{(10)}$:

$$\begin{array}{rclcl}
 35 & = & 17 \cdot 2 & + & 1 \\
 17 & = & 8 \cdot 2 & + & 1 \\
 8 & = & 4 \cdot 2 & + & 0 \\
 4 & = & 2 \cdot 2 & + & 0 \\
 2 & = & 1 \cdot 2 & + & 0 \\
 1 & = & 0 \cdot 2 & + & 1
 \end{array}
 \begin{array}{c} \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \\ \uparrow \end{array}$$

És a dir, $35_{(10)} = 100011_{(2)}$.

- Pel que fa a la part fraccionària:

$$\begin{array}{rclcl}
 0,25 \cdot 2 & = & 0,5 & = & 0 + 0,5 \\
 0,5 \cdot 2 & = & 1 & = & 1 + 0
 \end{array}
 \begin{array}{c} \downarrow \\ \downarrow \end{array}$$

Així doncs, $0,25_{(10)} = 0,01_{(2)}$

- Finalment, s'ajunten totes les parts i obtenim $-35,25_{(10)} = 1100011,01_{(SM2)}$.

Per a representar exactament aquest nombre cal un format que tingui un total de 9 bits:

- Un bit de signe
- 6 bits per a la part entera
- 2 bits per a la part fraccionària

7. L'equivalència binària d'aquests nombres decimals és la següent:

$$+12,25_{(10)} = +1100,01_{(2)}$$

$$+32,5_{(10)} = +100000,1_{(2)}$$

En signe i magnitud i 9 bits en què 2 són fraccionaris, la codificació és la següent:

$$+1100,01_{(2)} = 0001100,01_{(SM2)}$$

$$+100000,1_{(2)} = 0100000,10_{(SM2)}$$

Per fer la suma examinem els bits de signe. Es tracta de dos nombres positius, per tant, s'han de sumar les magnituds i afegir al resultat 1 bit de signe positiu:

$$\begin{array}{r} 0 \ 0 \ 1 \ 1 \ 0 \ 0, \ 0 \ 1_{(2)} \\ + \quad 1 \ 0 \ 0 \ 0 \ 0 \ 0, \ 1 \ 0_{(2)} \\ \hline 1 \ 0 \ 1 \ 1 \ 0 \ 0, \ 1 \ 1_{(2)} \end{array}$$

En fer la suma no es produeix sobreeximent, perquè no hi ha ròssec en la darrera etapa, que és el que determina si hi ha sobreeximent en aquest format. Al resultat obtingut s'ha d'afegir el bit de signe per a tenir la codificació en signe i magnitud correcta: **0101100,11_(SM2)**.

8. Per a codificar en BCD tan sols hem de codificar cada dígit decimal amb 4 bits:

$$1 = 0001_{(2)}$$

$$7 = 0111_{(2)}$$

$$8 = 1000_{(2)}$$

Finalment, ajuntem els codis BCD per obtenir la cadena final: **000101111000**.

9. El format especificat té la forma donada per la taula següent:

Signe (s)	Exponent (e)	Mantissa
1 bit	4 bits	3 bits

- En primer lloc, hem de codificar la magnitud $12,346_{(10)}$ en base 2. En l'exercici d'autoavaluació 4 hem trobat que $12,346_{(10)} \approx 1100,010_{(2)}$.
- Seguidament normalitzem l'expressió obtinguda en la forma $\pm R \cdot 2^e$:
- $+12,346_{(10)} \approx +1,100010_{(2)} \cdot 2^3$.
- Finalment, obtenim els diferents elements que defineixen la representació:
 - Signe:** es tracta d'un nombre positiu, per tant, el bit de signe serà **0**.
 - Exponent:** L'exponent és $3_{(10)}$ que cal codificar en excés a $M = 2^q - 1$. Com que disposem de 4 bits, l'excés és $M = 2^3 = 8$. Per tant, cal representar $3_{(10)} + 8_{(10)} = 11_{(10)}$ amb 4 bits:

$$\begin{array}{rclcl} 11 & = & 5 \cdot 2 & + & 1 \\ 5 & = & 2 \cdot 2 & + & 1 \\ 2 & = & 1 \cdot 2 & + & 0 \\ 1 & = & 0 \cdot 2 & + & 1 \end{array} \quad \uparrow$$

És a dir, $e = 11_{(10)} = 1011_{(2)}$ (en excés a 8).

- Mantissa:** $R = 1,100010_{(2)}$. Com que hi ha bit implícit (1,X), només cal representar la part de la dreta de la coma i disposem de 3 bits (amb truncament). Així doncs, la mantissa seran els 3 bits a partir de la coma: **100**.
- Ara ja podem ajuntar totes les parts:

Signe (s)	Exponent (e)	Mantissa
0	1011	100

És a dir, la representació és **01011100**.

10. En primer lloc, desempaquetem la cadena de bits. Cada dígit hexadecimal correspon a 4 bits:

- 3 = 0011
- 7 = 0111
- 8 = 1000

Per tant, $378_{(16)}$ empaqueta la cadena de bits 001101111000. Aquesta cadena codifica un nombre en coma flotant amb 4 bits de mantissa: 001101111000. Cerquem el nombre decimal que representa:

- Bit de signe $s = 0$, per tant, es tracta d'un nombre positiu.
- Exponent: $0110111_{(2)}$, és a dir, $q = 7$. Per tant, l'excés serà $M = 2^{q-1} = 2^6 = 64_{(10)}$. Llavors, l'exponent val $e = 0110111_{(2)} - 64_{(10)} = 55 - 64 = -9_{(10)}$.
- Finalment, els bits que resten formen la mantissa que, com que està normalitzada en la forma $1,X$ amb bit implícit pren el valor $R = 1,1000_{(2)}$.

Ara ja podem calcular el nombre representat en base 10 aplicant la fórmula $\pm R \cdot 2^e$ i el TFN:

$$+1,1000_{(2)} \cdot 2^{-9} = +11_{(2)} \cdot 2^{-10} = +3 \cdot 2^{-10}_{(10)} = + \frac{3}{1024}$$

Glossari

arrel *f* Base d'un sistema de numeració.

base *f* Nombre de dígits diferents en un sistema de numeració.

BCD *m* Vegeu **decimal codificat en binari**.

binary coded decimal *m* Vegeu **decimal codificat en binari**.

binari *adj* Dit del sistema de numeració en base 2.

bit *m* Abreviació de *binary digit* ('dígít binari'). Correspon a la unitat d'informació més petita possible. Es defineix com la quantitat d'informació associada a la resposta d'una pregunta formulada d'una manera no ambigua, per a la qual només són possibles dues alternatives de resposta, i que, a més, tenen la mateixa probabilitat de ser escollides.

bit menys significatiu *m* Bit menys significatiu (més a la dreta) d'una representació numèrica posicional.
sigla **LSB**

bit més significatiu *m* Bit més significatiu (més a l'esquerra) d'una representació numèrica posicional.
sigla **MSB**

borrow *m* Ròssec emprat en l'operació de resta.

byte *m* Vegeu **octet**.

cadena *f* Conjunt d'elements d'un mateix tipus disposats un a continuació de l'altre.

carry *m* Vegeu **ròssec**.

coma fixa *f* Sistemes de representació numèrica que empen un nombre fix de dígits enters i fraccionaris.

coma flotant *f* Sistemes de representació numèrica que codifiquen un nombre variable de dígits enters i fraccionaris. La quantitat de dígits enters i fraccionaris depèn del format i del valor numèric representats.

decimal *adj* Dit del sistema de numeració en base 10.

decimal codificat en binari *m* Codificació dels dígits decimals (0, 1, 2, ..., 9) sobre un conjunt de 4 dígits binaris.
sigla **BCD**

dígit *m* Element d'informació que pot agafar un nombre finit de valors diferents. El nombre de valors diferents el determina la base de numeració.

digital *adj* Dit del sistema que treballa amb dades discretes.

empaquetament *m* Transformació que permet representar la informació binària de manera més compacta.

format *m* Descripció estructural d'una seqüència de dades, en la qual s'especifica el tipus, la llargada i la disposició de cada element.

fraccionari *adj* Més petit que la unitat.

hexadecimal *adj* Dit del sistema de numeració en base 16.

LSB *m* Vegeu bit menys significatiu.

mot *m* Unitat d'informació processada per un computador d'un sol cop (en paral·lel).
en word

MSB Vegeu bit més significatiu.

octal *adj* Dit del sistema de numeració en base 8.

octet *m* Cadena de 8 bits.

overflow *m* Vegeu sobreiximent.

precisió *f* Diferència entre dos valors consecutius d'una representació.

rang *m* Conjunt de valors que indiquen els marges entre els quals es troben els valors possibles d'un format de representació.

representació *f* Vegeu format.

sobreiximent *m* Indicació que el resultat d'una operació està fora del rang de representació disponible.
en overflow

ròssec Bit que indica que s'ha superat el valor *b* de la base de numeració en sumar dos dígit. La suma de dos valors numèrics es duu a terme dígit a dígit, de dreta a esquerra. Un ròssec en sumar dos dígit indica que cal afegir una unitat en fer la suma dels dos dígit següents.
en carry

word *m* Vegeu mot.

Bibliografia

De Miguel, P. (1996). *Fundamentos de los computadores* (5a ed., cap. 2). Madrid: Paraninfo.

Patterson, D.; Henessy, J. L. (1995). *Organización y diseño de computadores: la interfaz hardware/software*. Madrid: McGraw-Hill.

Stallings, W. (1996). *Organización y arquitectura de computadores: diseño para optimizar prestaciones*. Madrid: Prentice Hall.

