Fonaments de Computadors (05.562)

Universitat Oberta de Catalunya (UOC)

http://furniman.blogspot.com

Representació de la Informació

Circuits lògics combinacionals

Els circuits lògics seqüencials

Estructura bàsica d'un computador

Qüestionari

Representació de la Informació

1.- Els nombres i els sistemes de representació

1.1.- Sistemes de representació

- Un **sistema de numeració** és una metodologia que permet representar un conjunt de valors numèrics.
- 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 dígits diferents de què disposem.
- Els sistemes que fan servir només una base s'anomenen **sistemes de numeració de base fixa.** (sistema decimal)
- Els sistemes que fan servir més d'una base s'anomenen **sistemes de numeració de base mixta.** (Sistema horari)
- Els sistemes de numeració en els quals l'ordre dels dígits és determinant en la representació numèrica s'anomenen **sistemes posicionals.**

1.2.- Sistemes de numeració posicionals

- Un sistema de numeració posicional de base fixa és aquell en què un valor numèric X es representa com una seqüència ordenada de dígits de la manera següent:
  $x_{n-1}\, x_{n-2} \ldots x_1\, x_0, x_{-1} \ldots x_{-m}$ / $0 \le x_i \le b\text{-}1$
  - b és la base del sistema de numeració
  - x_i és el dígit de la posició i-èsima de la seqüència.
  - Les posicions amb subíndex positiu corresponen a la **part entera** del nombre.
  - Les posicions amb subíndex negatiu corresponen a la **part fraccionaria** del nombre.
  - La frontera entre la part entera i la part fraccionaria s'indica amb una **coma.**
  - Els dígits de la part entera es consignen a l'esquerra de la coma i els de la part fraccionaria a la dreta.
- El sistema de base 10 rep el nom de **decimal.**
- El sistema de base 16 s'anomena **hexadecimal.**
- El sistema de base 8 s'anomena **octal.**
- El sistema de base 2 s'anomena **binari.**

Teorema fonamental de la numeració (TFN)

- $x = \sum_{i=-m}^{n-1} x_i b^i = x_{n-1}b^{n-1} + x_{n-2}b^{n-2} + \cdots + x_{-m}b^{-m}$ / $0 \le x_i \le b\text{-}1$

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	110	22	18	12

1.3.- Canvis de base

1.3.1.- Canvi de base segons el TFN

- $x_2\, x_1\, x_0, x_{-1}\, x_{-2}\,{}_{(b} = x_2 b^2 + x_1 b^1 + x_0 b^0 + x_{-1} b^{-1} + x_{-2} b^{-2}$
  - Serveix per a passar a base 10.
  - Per a b>10, els dígits de la base b s'han de canviar a base 10 abans de fer les operacions.

$462_{(7} = 4\cdot7^2 + 6\cdot7^1 + 2\cdot7^0 = 240_{(10}$

$101100{,}01_{(2} = 1\cdot2^5 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^{-2} = 44{,}25_{(10}$

$AF2C_{(16} = 10\cdot16^{3} + 15\cdot16^2 + 2\cdot16 + 12$

1.3.2.- Mètode basat en el teorema de la divisió entera

- Pa sa de base 10 a base b
- Per trobar la part entera:
  - Dividim successivament entre b fins a obtenir un quocient 0.
  - Els residus agafats de l'últim al primer formaran la part entera de la nova base.
  - Si b > 10 els residus s'han de passar a dígits de la nova base
- Per trobar la part fraccionaria
  - S'agafa la part fraccionaria i es multiplica per la base
  - La part entera serà un dels dígits cercats.
  - Tornem a agafar la part fraccionaria i repetim el procés fins a trobar un comportament periòdic o tinguem suficients dígits.
  - Els nombres s'agafen del primer a l'últim.

Canviar a base 16 el $44844'12_{(10}$

44844=2802 * 16 + 12 (C) ↑
2802 = 175 * 16 + 2
175 = 10 * 16 + 15 (F)
10 = 0 * 16 + 10 (A)

0'12 * 16 = 0'92 + 1 ↓
0'92 * 16 = 0'72 + 14 (E)
0'72* 16 = 0'52 + 11 (B)
…

$44844'12_{(10} = AF2C'1EB\ldots$

Canvi entre dues bases <> 10

- Farem servir el mètode TFN per passar a base 10 i després el mètode de la divisió entera per passar a la nova base.

1.3.3. – Canvi de base entre b i b^n

Canvi de base b a base b^n

- S'agafaran grups d'n caràcters. $(16 = 4^2 = 2^4)$
- Es partirà sempre de la coma i s'ompliran els dígits que manquen amb zeros.

Canvi de base b^n a base b

- El mateix a l'inrevés

Passar $101110'101101_{(2}$ a base 16

0010 1110 , 1011 0100
2 E B 4

Passar $7672'13_{(8}$ a base 2

7 6 3 2 ' 1 3
111 110 011 010 ' 001 011

1.4.- Empaquetament de la informació

- Consisteix en agafar grups de 4 dígits binaris i representar-los amb el valor hexadecimal corresponent.
- Al final de la cadena empaquetada posem una *h* per indicar que és un valor hexadecimal.
- El procés contrari s'anomena **desempaquetament.**
- Quan empaquetem prescindim de les comes i si falta algun dígit el posem a l'esquerra abans d'empaquetar.

1.6. / 1.7. – Suma i resta en els sistemes posicionals

Sigui quina sigui la base, la forma d'operar és com en base 10

```
    1                ← ròssecs
    3  5  8  2 (16
+   A  F  1  8 (16
-------------------
    E  4  9  A (16   ← resultat
```

```
    1 0 1 0 1 1 0 1 (2
    1 1 1           ← ròssecs
-   0 1 1 1 0 1 0 0 (2
-------------------
    0 0 1 1 1 0 0 1 (2   ← resultat
```

- En anglès el ròssec de la suma s'anomena *carry*
- En anglès el ròssec de la resta s'anomena *borrow*

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

- **Multiplicar per b^k** un nombre en un sistema posicional de base fixa *b* equival a desplaçar la coma fraccionaria k posicions a la dreta.
- **Dividir per b^k** un nombre en un sistema posicional de base fixa *b* equival a desplaçar la coma fraccionaria k posicions a l'esquerra.

$11010_{(2} * 2^4 = 110100000_{(2}$

$11100_{(2} / 2^4 = 1'11_{(2}$

Quocient i residu

- El **quocient** d'una divisió entera d'un nombre enter per una potencia de la base de numeració correspon a la part entera de la divisió real.
- El **residu** serà la part fraccionaria multiplicada pel divisor.

$11100_{(2} / 2^4 = 1'11_{(2}$

Quocient => $1_{(2}$
Residu => $0'11_{(2} * 2^4 = 1100_{(2}$

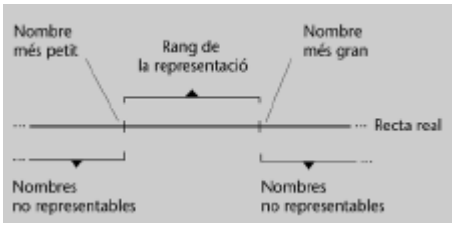
2. Representació dels nombres en un computador

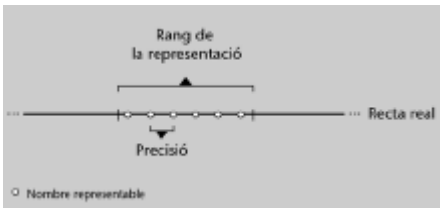
2.1.- Condicionants físics

- **Error de representació**
  - Quan un nombre no es pot representar de manera exacta dins d'un computador.
  - És la distància entre el nombre que volem representar i el nombre representat realment.
- **Format de representació**
  - 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.

2.1.1.- Rang de representació

- 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 sevir és [a,b], on *a* i *b* són els límits de l'interval i en formen part.
- Amb *n* dígits en base *b* es poden representar un màxim de b^n nombres diferents.





2.1.2.- Precisió

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

2.1.3.- Error de representació

- $\varepsilon = |X - \hat{X}|$
- És la distància entre el nombre X que volem representar i el nombre representable $\hat{X}$ al qual l'aproximem
- Els nombres que no són dins del rang de representació del format no són *representables* ni *aproximables*.

2.1.4. Aproximacions: truncament i arrodoniment

- **Truncament**
  - Consisteix en menysprear els dígits fraccionaris que no caben en el format
  - No comporta cap càlcul
  - L'**error màxim** de representació és inferior a la precisió del format de representació.
- **Arrodoniment**
  - També s'anomena quantificació
  - Consisteix en escollir el nombre representable més proper al que volem representar.
  - L'**error màxim** de representació és igual a la meitat de la precisió del format de representació.
  - Comporta operacions aritmètiques.
    - Sumem la meitat de la precisió del format de representació al nombre que volem representar.
    - Trunquem el resultat de la suma segons el nombre de dígits fraccionaris disponibles al format de representació.

2.1.5.- Sobreeiximent

- Quan el resultat d'una operació supera el rang de representació
- En anglès s'anomena *overflow*
- El **sobreeiximent a zero** apareix en un nombre de magnitud menor que la precisió del format que s'acaba representant com a zero. (En anglès *underflow*)

2.2.- Nombres naturals

- No tenen part fraccionaria ni signe (0,1,2,3…)
- El **rang** de representació dels nombres naturals en un format d'n bits és en decimal $[0, 2^n - 1]$ i la seva **precisió** és 1.
- Ampliar el nombre de bits d'un format de representació s'anomena **extensió** i s'aconsegueix afegint zeros a l'esquerra fins a completar els *m* bits del format nou.
- **Sobreeiximent**
  - En la suma de dos nombres naturals es produeix quan tenim un ròssec a la darrera etapa de suma.
  - L'operació de resta no pot donar lloc a sobreeiximent.
  - En la multiplicació hi ha sobreeiximent si el resultat supera el rang del format.
  - La divisió entera per una potència de la base no produeix sobreeiximent, per què el resultat són dos nombres naturals (*quocient* o *residu*)

2.3.- Nombres enters

- Son nombres amb signe sense part fraccionària incloent el 0. (…,-3,-2,-1,0,1,2,3…)
- Es diferencien dels naturals per un signe que indica si la magnitud és positiva o negativa.

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

- El bit més significatiu (MSB) emmagatzema el signe i la resta la magnitud.
- Un 1 en el bit més significatiu indica signe negatiu, un 0 indica signe positiu.
- Farem servir la notació $X_{(SM2}$ per identificar un nombre codificat en signe i magnitud en base 2.
- En general, en SM2, el **rang** d'enters representable amb *n* bits és, en decimal $[-(2^{n-1}-1), 2^{n-1}-1]$
- L'**extensió** s'aconsegueix afegint zeros a l'esquerra de la magnitud fins a completar els *m* bits, mantenint el MSB.

2.3.2.- Suma i resta en signe i magnitud.

- Els inconvenients d'aquest sistema son la dificultat de les operacions i l'existència de dos signes per al zero.
- Per operar eliminarem el dígit de signe i decidirem quin posarem al final.
- **Suma mateix signe:** Se sumen normalment i es posa el signe que porten. *Pot tenir sobreeiximent*.
- **Suma signe diferent:** Es resta la magnitud petita de la gran i s'aplica el signe de la gran. *No hi ha sobreeiximent*.
- **Resta mateix signe:** Es resta la magnitud petita de la gran i es posa el signe que porten. *No hi ha sobreeiximent.*
- **Resta diferent signe:** Se sumen normalment i si no hi ha sobreeiximent es posa un 1 com a signe.

2.3.3.- Representació en complement a 2

- S'abreuja Ca2 o C2. Nosaltres farem servir $X_{(Ca2}$ per referir-nos a un nombre codificat en complement a 2.
- Actualment és el sistema més emprat per a codificar enters als ordinadors ja que té una codificació única pel zero i simplifica les operacions de suma i resta.
- El **nombres positius** es codifiquen igual que a signe i magnitud. (El MSB és un zero i la resta la magnitud)
- Per a representar un nombre negatiu (-X) en Ca2, resolem $2^n - |X|$.
  - $|X|$ és el valor absolut d'X
  - n és el nombre de dígits amb que treballem.
  - 2^n en binari realment és un 1 seguit d'n zeros.
- També podem fer el complement a 2 (veure punt següent) de la magnitud positiva i acabem abans.
- El **rang** d'enters representable amb *n* bits és, en decimal $[-2^{n-1}, 2^{n-1}-1]$
- L'**extensió** s'aconsegueix afegint zeros a l'esquerra per als nombres positius i uns per als negatius.

2.3.4.- Canvi de signe en complement a 2

- Serveix tant per passar de magnituds positives a negatives com a l'inrevés.
- Mètode a)
  - Canviem 1 per 0 i 0 per 1 excepte a l'últim nombre.
- Mètode b)
  - De dreta a esquerra mantenim els dígits que trobem fins al primer 1 (aquest inclòs)
  - Fem el complement de la resta (0 per 1 i 1 per 0)

2.3.5.- Magnitud dels nombres en complement a 2.

- Per a nombres positius fem servir el TFN
- Per als negatius:
  - Apliquem TFN però multipliquem el MSB per -1
  - O bé, fem un canvi de signe i calculem per MSB. (Tenint en compte que ens sortirà la magnitud positiva.)

2.3.6.- Suma en complement a 2

- Se suma normal sense eliminar el dígit de signe.
- El ròssec a l'última etapa **NO** indica sobreeiximent. Simplement es menysprea.
- Si els dos valor son de mateix signe però el resultat té signe canviat, vol dir que hi ha **sobreeiximent**.
- Si els dos valors son de signe diferent, no hi haurà mai sobreeiximent, si hi ha ròssec a l'última etapa, simplement es menyspreará.

2.3.7.- Resta en complement a 2

- Es canvia el signe al subtrahend i se suma.

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

- S'afegeixen k zeros a la dreta
- Si treballem amb n bits, si es perd un 1 per a nombres positius, un 0 per a nombres negatius o el resultat canvia de signe, parlarem de **sobreeiximent**.

2.4.- Nombres fraccionaris

- Son els que tenen una part més petita que la unitat.

Representació binària en coma fixa

- La coma no s'especifica, és a la definició del format on s'especifica quines posicions seran les decimals.
- En coma fixa, el més habitual és treballar amb signe i magnitud.

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

- Separem el MSB que ens indicarà el signe
- Apliquem TFN per trobar la conversió a decimal

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

- Canviem a base 2 amb el teorema de la divisió entera.
- Si la part fraccionaria excedeix el nombre de bits disponibles, haurem de truncar o arrodonir.
    - Per arrodonir sumem la meitat de la precisió, és a dir 0'xxx1 sent x tants zeros com la precisió i després trunquem.

Rang i precisió en coma fixa.

- Des de tot uns (negatiu) fina tots uns menys el MSB que serà zero (positius)
    - Amb signe i magnitud: $[-2^{n-m-1} + 2^{-m}, +2^{n-m-1} - 2^{-m}]$
    - Sense signe $[0, +2^{n-m}-2^{-m}]$

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

- S'aconsegueix afegint x zeros per l'esquerra a la part entera (mantenint el primer dígit que indica si és negatiu) i z zeros per la dreta.

Precisió d'un format de coma fixa.

- 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.

- Se suma i resta com enters i es manté la coma
- En la suma de dos nombres sense signe en coma fixa, el **sobreeiximent** es produeix quan tenim un ròssec a la darrera etapa de suma. (com als nombres naturals)
- L'operació de resta de dos nombres sense signe en coma fixa, no pot donar lloc a **sobreeiximent**. (com els naturals)
- El **sobreeiximent** en la suma i resta de dos nombres en coma fixa i signe i magnitud, és pot produir si en la suma de dos nombres d'igual signe apareix un ròssec a la darrera etapa. (com als enters en signe i magnitud)
- Restar la magnitud gran de la petita per a nombres sense signe no és una operació vàlida (El resultat portaria signe).

Multiplicació i divisió per 2^k en coma fixa binaria

- Multiplicar per 2^k un nombre en coma fixa sense signe equival a desplaçar els bits k posicions a l'esquerra i completar els n bits dels format afegint zeros.
- Dividir per 2^k un nombre en coma fixa sense signe equival a desplaçar els bits k posicions a la dreta i completar els n bits dels format afegint zeros per l'esquerra.
- Es produeix **sobreeiximent** en multiplicar un nombre sense signe per 2^k quan es perden 1 o més bits significatius (1) en desplaçar els bits *k* posicions.
- La divisió no produeix **sobreeiximent**.

3.- Altres tipus de representacions

3.1.- Representació d'informació alfanumèrica.

- És la informació no numèrica constituïda pel conjunt de lletres, xifres i símbols que es fan servir a les descripcions textuals i que reben el nom genèric de *caràcters*.
- La codificació ASCII *(American standard code for information interchange)* és la més estesa per codificar caràcters
  - Té una versió bàsica de 128 símbols que forma l'estàndard i una versió estesa no tant estandarditzada de 256.
  - La versió bàsica es codifica amb 7 bits i la estesa amb 8 bits.
  - Els primers 31 codis i el darrer són caràcters de control utilitzats per donar format al text o per controlar perifèrics.
  - El caràcter 32 (SP) representa l'espai en blanc, altres caràcters de control no visualitzable son:

d \ u	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		

DEL	Esborrar	**ESC**	Escapada	**HT**	Tabulador horitzontal
LF	Final de línia	**CR**	Retorn a primera columna	**FF**	Final de pàgina
STX	Inici de text	**ETX**	Final de text		

- L'estàndard *Unicode* te 16 bits i inclou la codificació de text en més d'una llengua.
- Els codis ASCII de 8 bits dels caràcters visibles es poden convertir a Unicode si afegim 8 zeros a l'esquerra.

3.2.- Codificació de senyals analògics

- Un senyal elèctric analògic és aquell que codifica la infirmació mitjançant una variació d'un paràmetre elèctric (tensió, freqüència, intensitat) que s'ajusta de manera proporcional a l'estímul original.
- El procés de convertir una representació analògica a una digital s'anomena **digitalització.**
- La digitalització consta de tres etapes:
  - **Mostratge (o discretització):** Consisteix a prendre mostres del senyal analògic a intervals de temps regulars.
  - La **quantificació** consisteix a assignar un valor, d'entre un conjunt finit, a l'amplitud del senyal en cada interval de mostartge.
  - **Codificació binaria:** Consisteix en expressar els valors de les mostres amb zeros i uns.

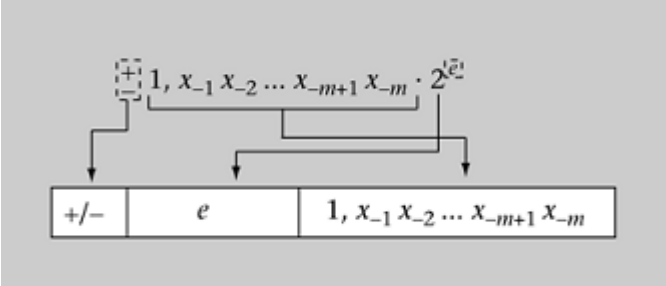
3.3.- Altres representacions numèriques

3.3.1.- Representació en excés a M

- Consisteix a convertir un nombre enter X en un natural sumant un desplaçament M al valor numèric X.
- Trobem el valor d'un nombre codificat en excés a M si restem M a la codificació.
- L'excés a *M* és un tipus de nombres enters emprada per a codificar el valor de l'exponent quan treballem en coma flotant.

3.3.2.- Representació en coma flotant

- També anomenat *notació científica*.
- Serveix per expressar nombres molt grans o molt petits amb pocs dígits.
- Pren la forma $\pm R \cdot b^e$
  - $\pm$ indica el signe (S) de la magnitud representada.
  - R és un nombre fraccionari que rep el nombre de **mantissa** (R)
  - b és la **base** de numeració.
  - e és un nombre enter que rep el nom d'**exponent** (e)**.** Indica el nombre de posicions a la dreta (exponent positiu) o a l'esquerra (exponent negatiu) que hem de desplaçar la coma fraccionaria de la *mantissa* per obtenir el valor numèric representat.
- En els computadors assumim que la **base de numeració** és dos i el nombre de bits de la mantissa i el nombre de bits de l'exponent el fixa el format.
- Les posicions més habituals en las que es fixa la coma de la mantissa són l'esquerra del primer dígit no nul i especialment, la dreta del primer dígit no nul.



- En coma flotant, l'exponent sol estar restringit als enters. Per a codificar-lo, el més habitual és fer servir excés a M que pren els valors 2^{q-1} o 2^{q-1}-1en els quals q és el nombre de bits de l'exponent.
- Com la mantissa en format normalitzat sempre serà 1 la podem eliminar amb el que guanyarem un bit. S'anomena tècnica del **bit implícit.**

Descodificació d'un nombre en coma flotant

1. Identifiquem signe, base i mantissa.
2. Tenim en compte si la base té bit implícit o no.
3. L'exponent (e) serà:
   Valor de l'exponent – $2^{\text{digits de la mantissa} - 1}$
4. El resultat serà (+/-)R·2^e

3.3.3. Representació BCD (*binary coded decimal*)

- Consisteix a codificar cada dígit amb la seva correspondència de 4 bits, com si fossin caràcters.

Circuits lògics combinacionals

1.- Fonaments de l'electrònica digital

1.1.- Circuits, senyals i funcions lògiques

- Un **circuit** és un sistema format un conjunt de *dispositius electrònics* que operen sobre un cert nombre de *senyals d'entrada* generant un cert nombre de *senyals de sortida.*
- Els circuits poden prendre dos valors 1 i 0 en funció de si existeix tensió alta o baixa en els cables. S'anomenen circuits lògics o binaris.

1.2.- Àlgebra de Boole

- És una entitat matemàtica formada per dos elements, unes *operacions bàsiques* sobre aquests elements si una llista d'*axiomes* que defineixen les *propietats* que compleixen les operacions.
- Una variable booleana o **variable lògica** pot prendre els valors 0 i 1.
- Les operacions bàsiques són:
  - La **negació**, o complementació o NOT.
    - Correspon a la partícula NO
    - Es representa amb una cometa simple (')
    - Denota la negació de la variable.
  - El **producte lògic** o **AND**.
    - Correspon a la conjunció 'i' de la lògica
    - Es representa amb el símbol (·)
    - Denota el producte de dos variables
  - La **suma lògica** o **OR**
    - Correspon a la conjunció lògica ' o '
    - Es representa pel símbol (+)
    - Denota la suma lògica de dues variables

x	x'
0	1
1	0

Operació NOT

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Operació AND

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

Operació OR

- Axiomes
  - **Propietat commutativa:**
    - **x + y = y + x**
    - **x** · y = y · x
  - **Propietat associativa:**
    - x + (y + z) = (x + y) + z
    - x · (y · z) = (x · y) · z
  - **Propietat distributiva:**
    - **x** · (y + z) = x · y + x · z
    - x + (y · z) = (x + y) · (x + z)
  - **Element neutre**
    - **x + 0 = x**
    - **x** · 1 = x
  - **Complementació**
    - **x + x' = 1**
    - **x** · x' = 0

Teoremes de l'Àlgebra de Boole

1. **Principi de Dualitat**
   Tota identitat deduïda a partir dels axiomes continua essent certa si les operacions + i · i els elements 0 i 1 s'intercanvien en tota l'expressió.
2. **Llei d'idempotència**
   - x + x = x
   - x · x = x
3. **Llei d'absorció**
   - x + x · y = x
   - x · (x + y) = x
4. **Llei de dominància**
   - x + 1 = 1
   - x · 0 = 0
5. **Llei d'involució**
   - (x')' = x
6. **Lleis de Morgan**
   - (x+y)' = x'· y'
   - (x·y)' = x' + y'

1.3. – Representació de funcions lògiques

1.3.1.- Expressions algebraiques

- Estan formades per variables lògiques, els elements 0 i 1, els operadors producte (·), suma (+) i negació (‘), i els símbols (,) i (=)
- Una mateixa funció lògica es pot expressar mitjançant infinites expressions algebraiques equivalents.

1.3.2.- Taules de la veritat

- Expressa una funció lògica especificant el valor que té la funció per a cada possible combinació de valors de les variables d'entrada. Veure exemple a l'apartat d'Àlgebra de Boole
- Les propietats s'escriuran en ordre *lexicogràfic.* (0,0) (0,1) (1,0) (1,1)
- L'ordre de les columnes és significatiu. (A l'esquerra es posarà la variable de més pes)

1.3.3.- Correspondència entre expressions algebraiques i taules de veritat

- Passar una expressió algebraica a taules de veritat pot servir per a comprovar si dues expressions són equivalents.

1.3.4.- Expressions en suma de mintermes

- S'anomena *terme mínim* o **minterme** a una funció producte que només val 1 per a una sola combinació de variables d'entrada.
- S'aconsegueix fent agafant la combinació que dona com a resultat 1, i fent el producte de les variables que són zero negades per les variables que són 1.
- Qualsevol funció es pot expressar com a suma de mintermes. S'anomena *forma canònica* de la funció. (i als mintermes *termes canònics)*

a	b	a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

Funció O-exclusiva o XOR

1.4.- Altres funcions comunes

Funció O-exclusiva o XOR

- Val 1 quan alguna de les dos funcions d'entrada val 1, però NO quan valen 1 les dues alhora.
- Es representa amb el símbol ⊕
- a ⊕ b = a'b + ab'
- Es pot fer servir per saber si dos variables són iguals (sortida 0)
- Altres propietats:
  - 0 ⊕ a = a
  - 1⊕ a = a'

a	b	(a·b)'
0	0	1
0	1	1
1	0	1
1	1	0

Funció NAND

a	b	(a+b)'
0	0	1
0	1	0
1	0	0
1	1	0

Funció NOR

Funció NAND

- És la negació de l'AND: a NAND b = $(a \cdot b)'$
- Val 1 sempre que les dues variables d'entrada NO valguin 1

Funció NOR

- És la negació de l'OR: a NOR b = $(a + b)'$
- Val 1 quan cap de les variables d'entrada val 1 (O les dos valguin 0)

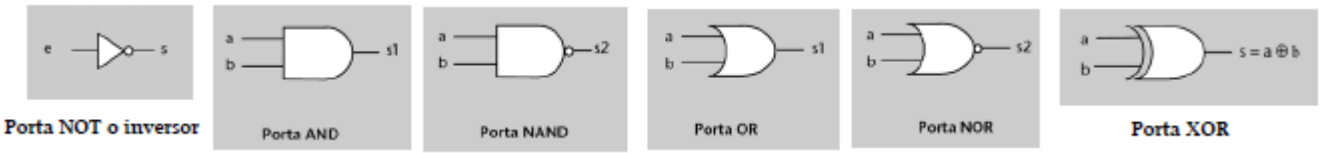
1.5.- Funcions especificades incompletament

- Les combinacions que, per les característiques del que representem, no es produiran mai, s'anomenen **combinacions no importa** o **combinacions don't care.**
- A una taula de veritat aquesta sortida es representa amb una x.

2.- Implementació de circuits lògics no combinacionals

2.1.- Portes lògiques. Síntesi i anàlisi

- Els dispositius electrònics que calculen funcions lògiques s'anomenen **portes lògiques.**
- Estan connectats a un cert nombre de senyals d'entrada (a l'esquerra) i una senyal de sortida (a la dreta)
- Totes les portes menys la NOT poden tenir n senyals d'entrada
  - Una porta AND d'n entrades val 1 només quan les n estrades són 1.
  - Una porta OR d'n entrades val 1 quan una o més de les n entrades són 1.
  - Una porta XOR d'n entrades val 1 quan hi ha un nombre senar d'entrades a 1. Si tot és zero val 0.

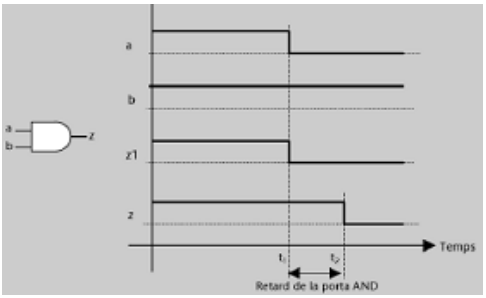


Síntesi i anàlisi

- S'anomena **síntesi** (o implementació) al procés d'obtenir un circuit a partir d'una expressió algebraica.
  - N'hi ha prou amb substituir cada operador de la funció per la porta lògica adeqüada.
- S'anomena **anàlisi** a obtenir l'expressió d'una funció a partir del circuit que l'implementa.
  - Es van escrivint les expressions a la sortida de cada porta.

2.2.- Disseny de circuits a dos nivells

2.2.1.- Retards. Cronogrames. Nivells de portes.



- Les portes lògiques no responen instantàniament a les variacions en els senyals d'entrada. Tenen un **retard**.
- Un **cronograma** és una representació gràfica de l'evolució dels senyals d'un circuit al llarg del temps.
- En un cronograma les línies de punts horitzontals representen el valor lògic 0 per a cada senyal.
- Les línies continues gruixudes representen el valor en que es troba cada senyal en cada moment.
- En vertical es poden assenyalar amb línies discontinues instants determinats de temps.
- El nombre de **nivells de portes** d'un circuit és el màxim nombre de portes que un senyal ha de travessar consecutivament per a generar un senyal de sortida. No es tenen en compte les portes **NOT.**

2.2.2- Síntesi a dos nivells

- Podem garantir que un circuit no tindrà més de dos nivells trobant la funció en suma de minterms.

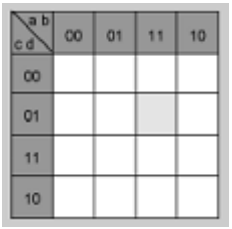
2.3- Minimització de funcions

2.3.1.- Simplificació d'expressions

- Si en dos mintermes totes les variables es mantenen constants llevat d'una, podem treure factor comú eliminant la variable que canvia.
  $f(x,y,z)=xy'z + xy'z' = xy' \cdot (z+z') = xy' \cdot 1 = xy'$
- En gral. Si en 2^m mintermes totes les variables es mantenen constants llevat d'm, podem treure factor comú m vegades i eliminar les m variables que canvien. En el terme producte resultant hi apareixeran n-m variables.

2.3.2.- Síntesi mínima a dos nivells. Mètode de Karnaugh

- És una mecànica senzilla per a detectar visualment els casos en que es pot minimitzar una expressió en suma de mintermes en 4 passes:

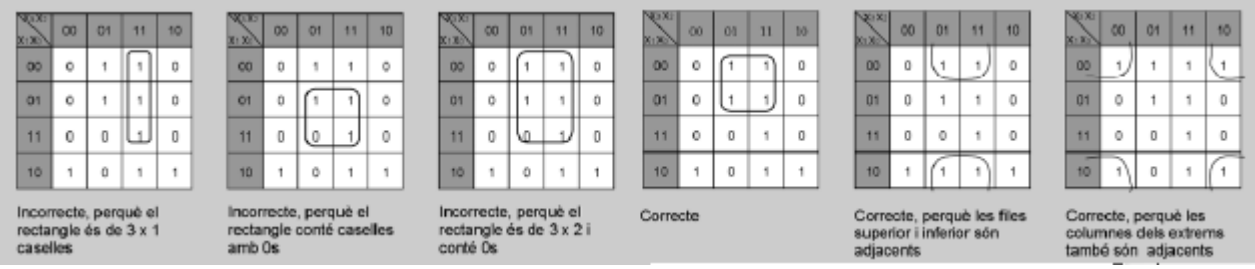


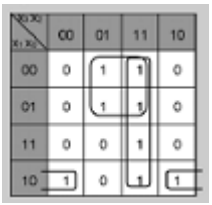
1. **Construcció del mapa de Karnaugh**

- És una transcripció de la taula de la veritat d'una funció a una estructura formada per caselles en la que cada casella correspon a una combinació de variables.
- Dues caselles del mapa són **adjacents** si corresponen a combinacions en las que només canvia el valor d'una variable.
- Les caselles NO estan en ordre lexicogràfic.

2. **Detecció dels casos en que es pot treure factor comú.**

- Consisteix en agrupar amb rectangles els grups d'uns adjacents, formant grups d'1, 2, 4, 8 o 16 uns.





- Tots els uns han de formar part d'algun grup.
- Els grups han de ser com més grans millor.
- Com més grups hi hagi, millor.
- Un mateix 1 pot formar part de més d'un grup si això ajuda a satisfer els dos objectius anteriors.

3. **Deducció del termes producte**

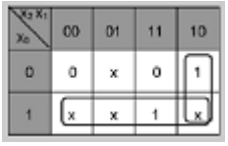
- Obtenim un terme producte de cada grup de la manera següent:
  - Només hi apareixen les variables el valor de les quals és constant per a totes les caselles que formen el grup.
  - Si en totes les caselles del grup una variable val 1, apareix en el terme del producte sense negar.
  - Si en totes les caselles del grup una variable val 0, apareix en el terme del producte negada.

4. **Obtenció de l'expressió mínima de la funció**

- S'expressa la funció com a suma lògica dels mintermes trobats al pas anterior.

2.3.3.- Minimització de les funcions especificades incompletament

- Al mapa de Karnaugh posarem una x a les caselles corresponents
- Farem que sigui 1 o 0 segons ens convingui per a obtenir el nombre més petit possible d'agrupacions i que siguin el més gran que puguem.



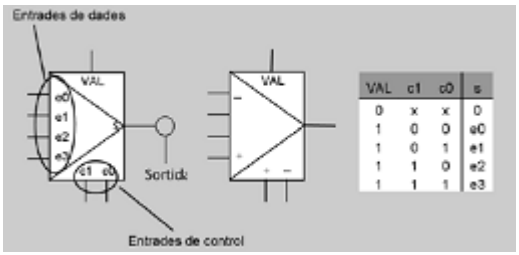
3.- Blocs Combinacionals

- És un circuit lògic combinacional amb una funcionalitat determinada. Està construït a partir de portes lògiques.

3.1.- Multiplexor. Multiplexor de busos. Demultiplexor.

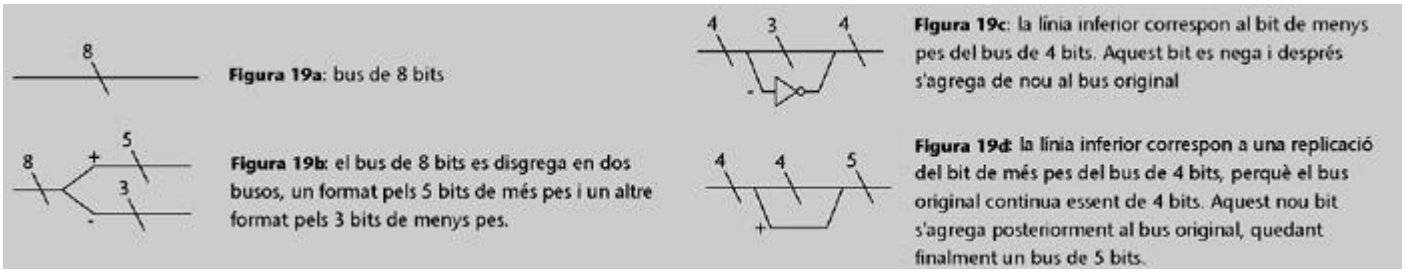
Multiplexor

- Un **multiplexor** és un bloc que fa la funció d'urbà en circuits electrònics.
- Consta de:
  - 2^m entrades de dades, identificades per la lletra *e* numerada desde 0 fins a 2^m-1 *(e0, e1, e2…)*
  - Uns sortida de dades *s*.
  - *m* entrades de *control* o *de selecció*, identificades per la lletra *c* numerada desde 0 fins a m-1 *(c0, c1, c2…)*
  - Una entrada de validació anomenada VAL
- S'identifica un multiplexor amb la nomenclatura 2^m-1. (Ex: 4-1)

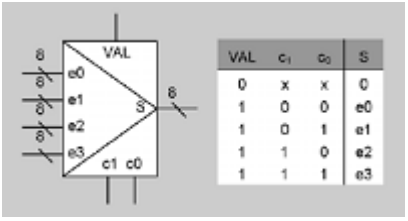


Multiplexor de busos

- Un **mot d'n bits** és una agrupació d'n bits.
- Els mots s'anomenen amb una lletra majúscula i els bits que el formen amb la lletra amb minúscula seguida d'un subindex que n'indica el pes: $A = a_7a_6a_5a_4a_3a_2a_1a_0$
- Un **bus** és una agrupació d'un cert nombre de cables, per cadascun dels quals circula un bit. Per tant un mot d'n bits circularà per un bus d'n bits.
- n és *l'amplada o mida* del bus.



- Un **multiplexor de busos** funciona igual que un multiplexor de bits, però en aquest cas les entrades de dades i la sortida són busos d'un determinat nombre de bits.

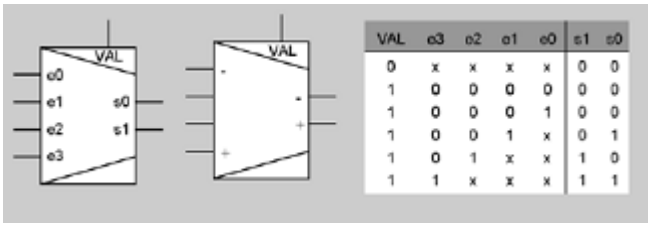


Desmultiplexor

- Fa la funció inversa d' un multiplexor.

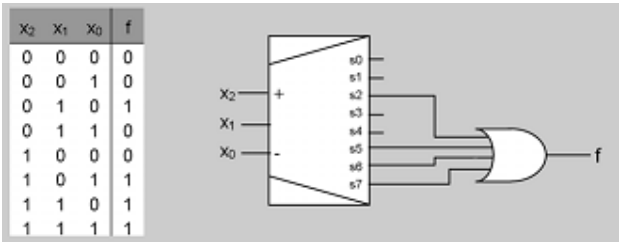
3.2.- Codificadors i descodificadors

- La funció d'un codificador és generar la codificació binaria d'un nombre.
- Té els senyals següents:
  - Una entrada de validació VAL, que al igual que els multiplexors si val 0 totes les sortides valen 0.
  - 2^m entrades de dades (d'u nbit), identificades per la lletra *e* numerada desde 0 fins a 2^m-1 *(e0, e1, e2…)*
  - *m* sortides de dades (d'un bit), identificades per la lletra *s* numerada desde 0 fins a m-1 *(s0, s1, s2…)*



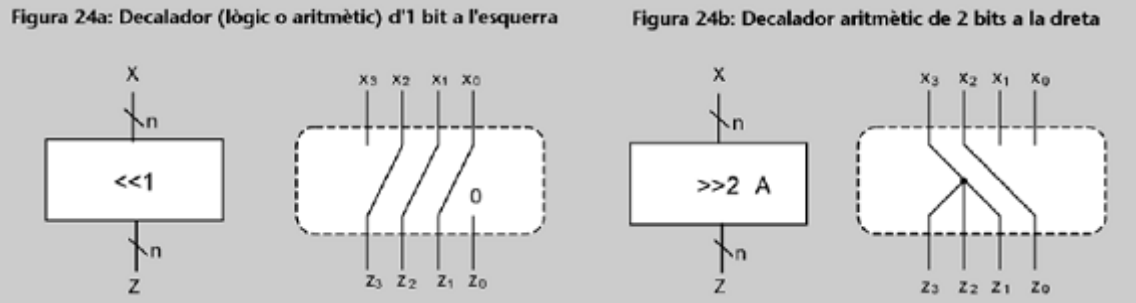
- Com a resultat dona el nombre de la senyal d'entrada de més pes = 1.
- El descodificador funciona a l'inversa.

Implementació d'una funció amb un descodificador



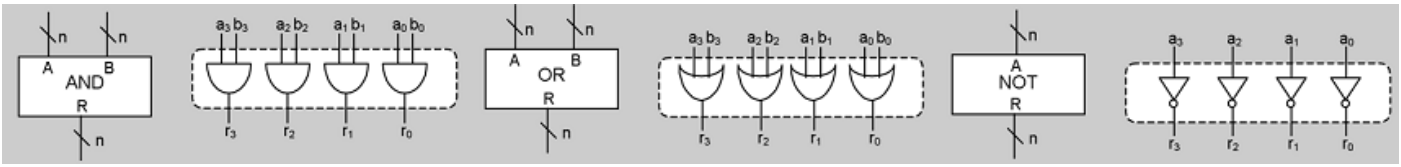
3.3.- Decaladors lògics i aritmètics

- Un **decalador** és un bloc combinacional que té la funció de desplaçar bits cap a l'esquerra o cap a la dreta.
- Tenen un senyal d'entrada i un de sortida tots dos d'n bits
- El senyal de sortida s'obté desplaçant els bits d'entrada *m* vegades cap a la dreta o l'esquerra.
  - Si el desplaçament és a l'esquerra, els bits de menys pes de la sortida es posen a 0 (equival a multiplicar per 2^m)
  - Si el desplaçament és a la dreta:
    - En els **decaladors lògics** es posen a 0. (equival a dividir per 2^m)
    - En els **decaladors aritmètics** prenen el valor del bit de més pes de l'entrada (equival a dividir per 2^m en complement a 2)



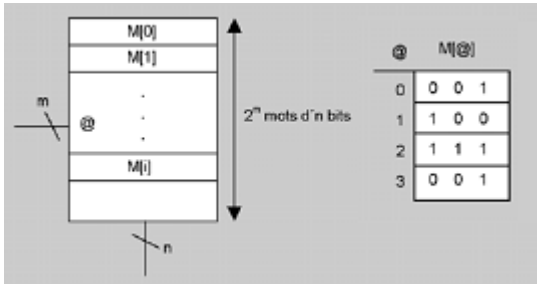
3.4.- Blocs AND, OR i NOT

- Fan les operacions AND, OR i NOT bit a bit sobre entrades d'n bits.
- Tenen dues entrades i sortides d'n bits (una en el cas del bloc NOT)
- Podem trobar blocs anàlegs com el XOR



3.5.- Memòria ROM

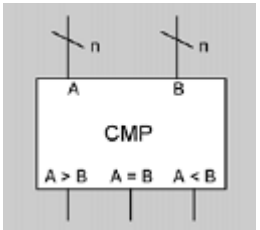
- És un bloc combinacional que permet guardar el valor de 2^m mots d' n bits.
- Consta de:
  - 2^m mots o dades d'n bits, cadascun a una posició diferent de la memòria. Les posicions es numeren des de 0 fins a 2^m-1 **(adreces)**
  - Una entrada d'adreces d'm bits identificada amb el símbol @.
  - Una sortida de dades d'n bits.



- Els bits d'entrada codifiquen una adreça de la memòria M[i] i es retorna el valor emmagatzemat en aquella adreça.

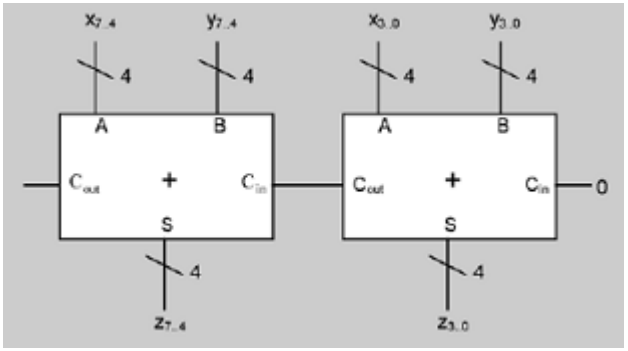
3.6.- Comparador

- Compara dos nombres codificats en binari i indica quina relació hi ha entre aquests.
- Consta de:
  - Dues entrades d'n bits que reben els noms A i B
  - Tres sortides d'un bit (A>B, A=B, A<B) on només una pot valer 1.



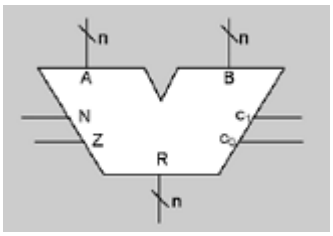
3.7.- Sumador

- Fa la suma de dos nombres codificats en binari o en complement a 2 (de fet la suma es fa igual)
- Consta de:
  - Dues entrades d'n bits anomenades A i B per on arribaran els nombres que s'han de sumar
  - Una sortida *S* que prendrà el valor de la suma d'A i B
  - Una sortida d'un bit C_{out}, que val 1 si es produeix un ròssec en el bit de més pes.
  - Una entrada d'un bit C_{in} per on arriba un bit d'entrada.
  - La sortida C_{out} es pot fer servir per sumar nombres de 2·n bits i només tenim sumadors d'n bits.
  - La sortida C_{out} indica sobreeiximent en entrades en binari, però no diu res en complement a 2.



3.8.- Unitat aritmètica i lògica (UAL)

- És un aparell capaç de fer un conjunt d'operacions aritmètiques i lògiques sobre dos nombres d'entrada codificats en binari o en C2
- Consta de:
  - Dues entrades d'n bits anomenades A i B
  - Una sortida R d'n bits amb el resultat de l'operació
  - Un cert nombre d'entrades de control c_i (m entrades per a 2^m operacions)
  - Un cert nombre de sortides d'un bit (**bits d'estat**) que indiquen circumstàncies en el càlcul
    - C indica si ha hagut ròssec en l'últim bit
    - V si hi ha sobreeiximent
    - N si el resultat és negatiu
    - Z si el resultat és 0



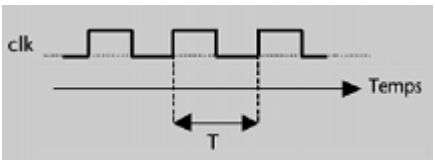
Els circuits lògics seqüencials

1.- Caracterització dels circuits lògics seqüencials

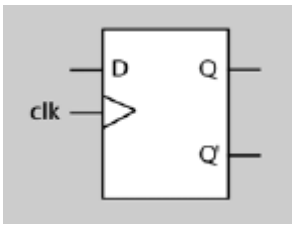
1.1.- Necessitat de memòria en els circuits lògics

- Els circuits lògics seqüencials es diferencien dels combinacionals perquè son capaços de recordar valors anterios d'alguns senyals.

1.2.- Rellotge. Sincronització.



- El **rellotge** és un senyal que determina en quin moment es fa la lectura del valor del senyal.
- La tasca que duu a terme el rellotge s'anomena **sincronització dels circuits.**
- El senyal del rellotge (*clk*) pren successivament els valors 0 i 1 en un interval de temps T anomenat **període, cicle o cicle de rellotge.**
- L'instant en que el senyal del rellotge passa de 0 a 1 s'anomena **flanc ascendent.**
- La freqüència és el nombre de cicles per segon i es mesura en *hertz*.

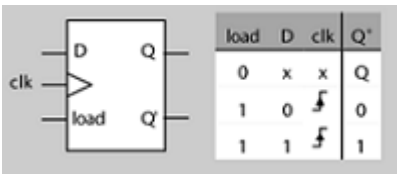


2.- El biestable D

2.1- Dispositiu elemental de memòria. El biestable D.

- Permeten guardar un bit de memòria
- Té dues sortides Q i Q'
  - Q és el valor que hi ha a l'entrada D a cada flanc ascendent del rellotge.
  - Q' és la seva negació

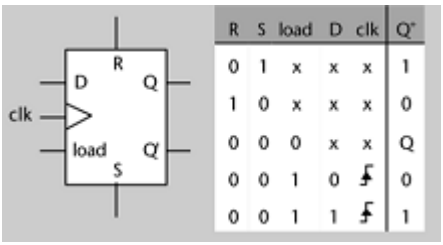
2.2- Senyals de càrrega



load	D	clk	Q+
0	x	x	Q
1	0	↓	0
1	1	↓	1

- Si a un biestable D li afegim un **senyal de càrrega** (load) funcionarà de a manera següent:
  - Si el load val 0, el valor del biestable no canvia (encara que canviï D)
  - Si el load val 1, funcionarà com a l'apartat anterior.

2.3- Entrades asíncrones

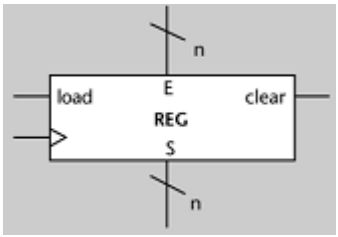


R	S	load	D	clk	Q+
0	1	x	x	x	1
1	0	x	x	x	0
0	0	0	x	x	Q
0	0	1	0	↓	0
0	0	1	1	↓	1

- Permeten modificar el valor del biestable independentment del valor dels altres senyals (té prioritat)
- Els biestables solen tenir dues entrades asíncrones:
  - R (*reset*): Quan es posa a 1, el biestable es posa a 0
  - S (set): Quan es posa a 1, el biestable es posa a 1
- Els circuits reals garanteixen que mai R i S poden ser 1 simultàniament.

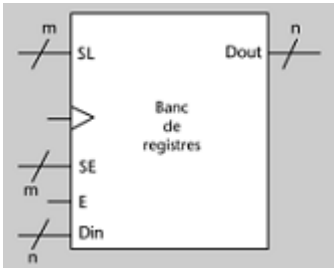
3.- Blocs seqüencials

3.1.- Registre



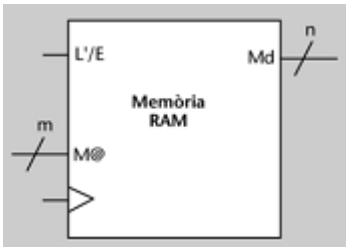
- És un bloc seqüencial format per n biestables D, que permet guardar el valor d'un mot d'n bits.
- Consta de:
  - Una entrada d'n bits. (E). Cada bit d'aquest bus es connecta a l'entrada D dels n biestables del registre. (**Escriptura** en el registre)
  - Una sortida d'n bits (S). És un bus format per les sortides Q dels n biestables del registre. (**Lectura** en el registre)
  - Dues entrades de control d'un bit *load* i *clear*, connectat al *load* i a *R* de cada biestable del registre.
  - Una entrada de rellotge connectada a les entrades de rellotge de tots els biestables.

3.2.- Banc de registres



- És una agrupació d'un cert nombre de registres tots del mateix nombre de bits.
- El nombre de registres d'un banc sempre és una potència de 2. Els registres en numeren des de 0 fins a 2^m-1
- Consta de:
  - Una entrada de selecció de lectura (SL) de m bits per a 2^m registres del banc.
  - Una sortida *Dout* de tants bits com registres del banc. *SL* i *Dout* formen el **port de lectura del banc.**
  - Una entrada de selecció d'escriptura *SE* d'*m* bits.
  - Una entrada de permís d'escriptura *E* d'un bit
  - Una entrada *Din* de tants bits com registres del banc. *SE* i *Din* formen el **port d'escriptura del banc.**
- Per a fer una lectura, a *SL* codifiquem el registre que volem llegir. El seu contingut estarà present a *Dout.* Immediatament, sense esperar al rellotge.
- Per a fer una escriptura, a SE codifiquem el registre en el que volem escriure i posem també l'entrada *E* a 1. Al proper flanc ascendent de rellotge s'escriurà al registre indicat per *SE* el valor qui hi hagi a *Din.*

3.3.- Memòria RAM



- És un bloc seqüencial que permet guardar 2^m mots d'n bits.
- És més gran i lent que un banc de registres.
- Consta de:
  - Una entrada d'adreces *M@*. Per a una capacitat de 2^m bits, tindrà *m* bits.
  - Una E/S *Md* dels mateixos bits que els mots que es guarden a la memòria.
  - Una entrada de control *L'/E* que indica si s'ha de fer una lectura o escriptura.
- Si L'/E = 0 es fa una lectura: Per *Md* surt el valor del mot guardat a l'adreça indicada per *M@*. Si *M@* canvia mentre L*'/E* = 0, *Md* canvia immediatament.
- Si L'/E = 1 es fa una escriptura: El mot indicat per *M@* pren el valor d'Md en el primer flanc de rellotge un cop activat L'/E. Quan *L'/*E = 1, el bus Md pren el valor 0.

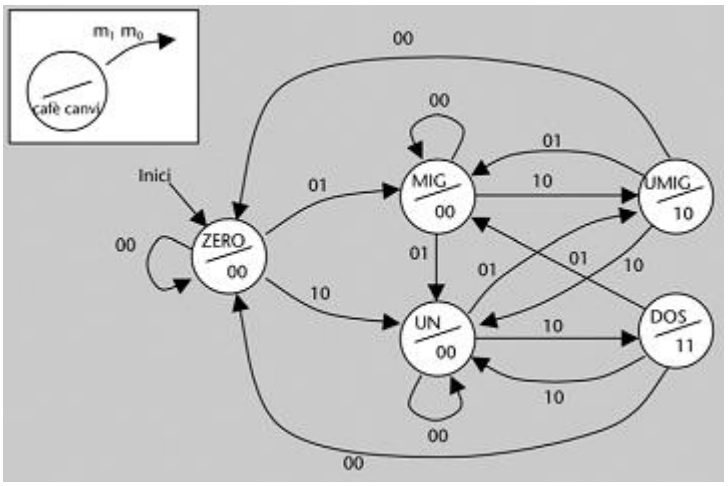
4.- El model de Moore.

4.1.- Estat. Transicions.

- El **model de Moore** és una forma d'expressar el funcionament d'un circuit lògic seqüencial. Es fonamenta en els conceptes d'*estat* i *transicions entre estats.*
- S'anomena **estat** cada situació diferent en què un circuit es por trobar.
- El valor dels senyals de sortida a cada estat s'anomena **taula de sortides.**
- L'estat en el que ens trobem s'anomena **estat actual**, i el proper estat, **estat futur.**
- El pas d'un estat a un altre s'anomena **transició** i s'especifiquen mitjançant una **taula de transicions.**
- Sempre tindrem un **estat inicial**
- En una taula de transicions es codifiquen totes les variables d'entrades possibles encara que no es produeixin mai. En aquest cas, el valor futur serà X (don't care)

4.2.- Representació gràfica: grafs d'estats.

- Un sistema seqüencial mitjançant el model de Moore, es representa mitjançant un **graf d'estats.**
  - Per a cada estat es dibuixa un cercle amb el nom de l'estat a la part superior. L'estat inicial se senyala amb una fletxa i la paraula *inici*.
  - A la part inferior s'escriu el valor dels senyals de sortida en aquest estat. Cal un **llegenda** que indiqui l'ordre dels senyals d'entada a les etiquetes dels arcs i els senyals de sortida en els cercles.
  - Les transicions es representen amb fletxes o **arcs.** La transició s'escriu al costat de l'arc.
  - No posarem els estat que no es donin mai.



4.3.- Sincronització

- Les transicions entre estats tenen lloc a cada flanc ascendent del rellotge.
- Si dibuixem les entrades en un cronograma, el valor és el que tenen en tocar *per l'esquerra*, la línea vertical corresponent a un flanc.

4.4.- Implementació

1. Codifiquem tots els estats amb una taula que necessita [$\log_2 n$] variables per codificar-los.
2. Codifiquem la taula de transicions o d'excitacions (és a dir estat origen – estat destí)
3. Codifiquem la taula de sortides
4. Ho posem tot en una taula

1

Estat	q_2	q_1	q_0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

2

Taula de transicions						
Estat				Estat+		
q_2	q_1	q_0	x	q_2^+	q_1^+	q_0^+
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	0
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	0	1	0	1	0
1	0	1	0	x	x	x
1	0	1	1	x	x	x
1	1	0	0	x	x	x
1	1	0	1	x	x	x
1	1	1	0	x	x	x
1	1	1	1	x	x	x

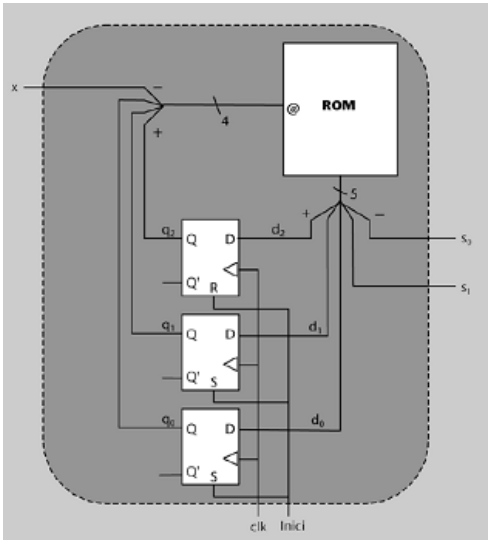
3

Taula de sortides				
Estat				
q_2	q_1	q_0	s_1	s_0
0	0	0	1	1
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	x	x
1	1	0	x	x
1	1	1	x	x

4

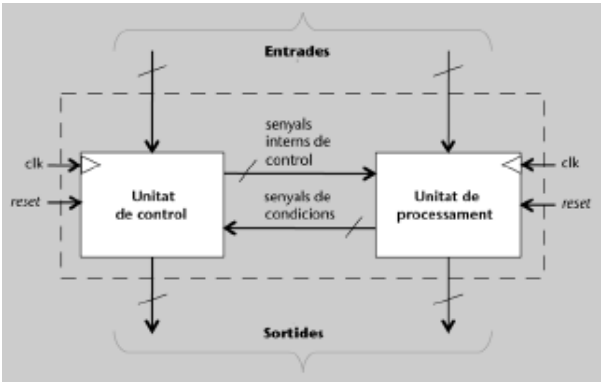
q_2	q_1	q_0	x	d_2	d_1	d_0	s_1	s_0
0	0	0	0	0	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	1	0	0	1	0	0	1
0	0	1	1	0	1	0	0	1
0	1	0	0	0	1	1	1	0
0	1	0	1	0	1	1	1	0
0	1	1	0	1	0	0	0	1
0	1	1	1	0	1	0	0	1
1	0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	1	1
1	0	1	0	x	x	x	x	x
1	0	1	1	x	x	x	x	x
1	1	0	0	x	x	x	x	x
1	1	0	1	x	x	x	x	x
1	1	1	0	x	x	x	x	x
1	1	1	1	x	x	x	x	x

5. Es representa amb una ROM



Estructura bàsica d'un computador

1.- Màquina d'estats



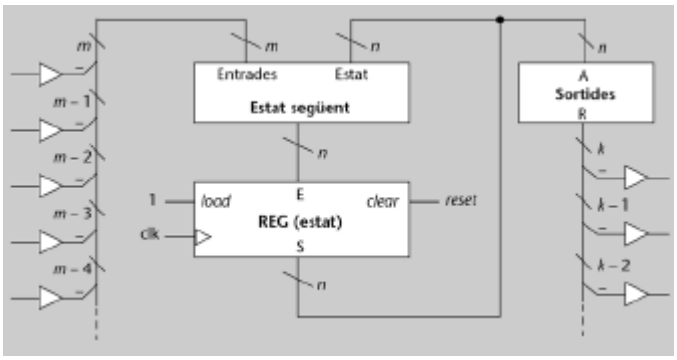
- Els circuits seqüencials es poden organitzar en dues parts:
  - La part que s'ocupa de les transicions d'estats i implementa les funcions d'excitació d'estats corresponent s'anomena **unitat de control.**
  - La part que s'ocupa de fer les operacions amb les dades per determinar condicions de transició com per calcular resultats de sortida s'anomena **unitat de processament.**

1.1.- Màquines d'estats finits com a controladors

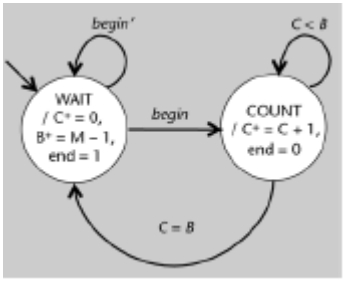
- Un **controlador** és una entitat amb capacitat d'actuar sobre un altra per portar-la a un estat determinat.
- Les **màquines d'estats finits (FSM)** són un model de representació del comportament de circuits molt adequat per als controladors.
- En un controlador tenim:
  - **estats captats** dels sistemes que controlen
  - sortides vinculades a l'estat del controlador (**accions**).

1.1.1- Procediment de materialització de controladors amb circuits seqüencials

1. D'acord amb la funcionalitat que s'ha d'implementar, decidir quines accions s'han de fer sobre el sistema que es controla. (Activar timbre / Apagar timbre)
2. Determinar les entrades del controlador. (Arriba un cotxe: SI / NO)
3. Dissenyar la màquina d'estats del controlador (graf)
4. Codificar en binari les entrades, els estats dels controladors i les sortides. (Taula de transicions i taula de sortida)
5. Dissenyar el circuit corresponent.



1.2.- Màquines d'estats finits esteses (EFSM)



- Poden combinar senyals de control d'un únic bit amb senyals de dades de més d'un bit.
- El canvi d'estat s'avalua a partir d'una condició (funció).
- Poden contenir variables
- En el graf els senyals de sortida es representen amb el valor que tenen en aquest mateix moment, mentre les variables s'anoten amb el valor que adquiriran a l'estat següent., és a dir que en un diagrama de temps, el contingut de les variables canvia en acabar l'estat actual.

La representació sol separar la part que s'ocupa de fer les operacions amb les dades numèriques amb les dades numèriques de la part que tracta amb els senyals d'1 bit i dels selectors. Aquesta arquitectura s'anomena **arquitectura de màquina d'estats amb camí de dades o FSMD.**

Materialització

1) A partir del graf d'estats s'obtenen les taules de transicions d'estats i de sortides:

Taula de transicions		
Estat Actual	**Entrada**	**Estat Futur**
Nom de l'estat	Valor de la fletxa	Nom de l'estat

Taula de sortides	
Estat Actual	**Sortida**
Nom de l'estat	Valors de *TOTES* les variables encara que no canviïn.

2) Si una variable pot fer dues operacions (Ex: B+ = B i B+ = M-1), s'haurà de codificar un selector:

Codificació de selectors		
Operació	**Selector**	**Efecte sobre la variable**
V+ = …	0,1 // 00,01,10,11…	Comentari

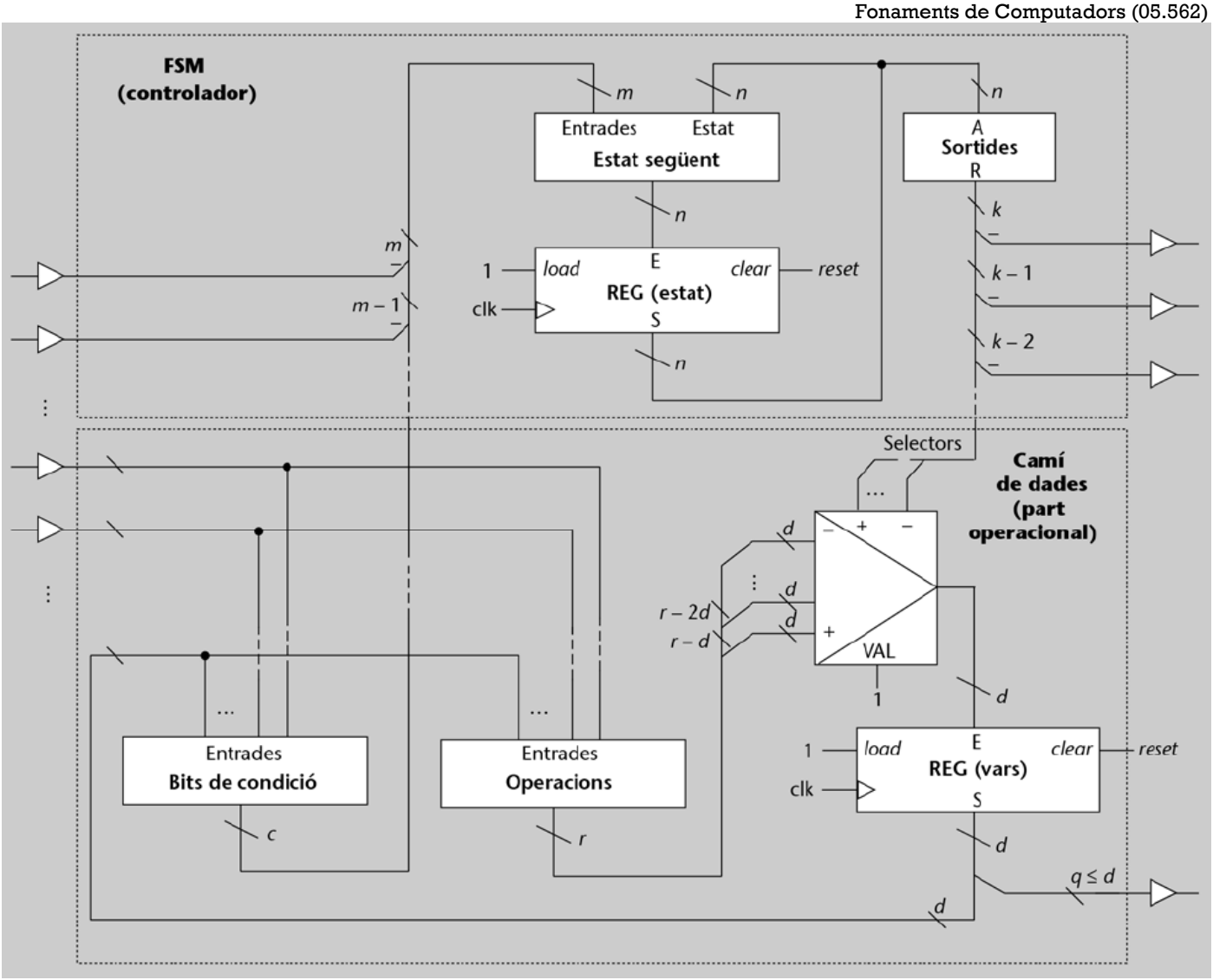
3) Fem la taula d'excitacions i de sortides:

Taula d'exitacions					
Estat actual		**Entrades**			**Estat+**
Símbol	**S**	**Contingut de les fletxes**	**Contingut de les fletxes**	**…**	**S+**
Nom de l'estat	Selector	0/1 si influeix en el canvi d'estat			Codificació de l'estat

Taula d'estats			
Estat	**Sortida 1**	**Sortida 2**	**Sortida n**
0,1 // 00,01,10,11…	(1 ó 0 si s'utilitza, x si no intervé)	(1 ó 0 si s'utilitza, x si no intervé)	(1 ó 0 si s'utilitza, x si no intervé)

4) Mirar al full següent l'arquitectura d'una FSM.

Al bloc *estat següent* es codificaran totes les variables on s+ sigui 1 com a suma de productes.



3.- Arquitectura bàsica d'un computador

3.1.- Màquina algorísmica general

- Una **màquina algorísmica general** interpreta les instruccions d'un programa emmagatzemat a la memòria.
- Un exemple és el *Femtoproc* que interpreta un repertori mínim d'instruccions:
  - Suma (ADD)
  - Operacions lògiques bit a bit (NOT i AND).
  - Salt si la darrera operació ha donat zero (JZ)
- En la materialització del *Femtoproc* farem servir una variable que contingui la posició de memòria de la instrucció a executar (PC o *program counter*)

3.2.- Màquina elemental

- S'anomena **llenguatge màquina** al llenguatge en què es codifiquen els programes que interpreta una màquina determinada.
- Una **microinstrucció** és una operació que es fa en la unitat de processament d'un processador en un cicle de rellotge.
- La **ALU** (unitat aritmeticològica) és un recurs de càlcul programable.
- En un *processador microprogramat* el **seqüenciador** fa de controlador d'una màquina d'interpretació d'instruccions (que no microinstruccions ja que també inclou la part del camí de dades corresponent que, entre d'altres coses, calcula el valor incrementat del comptador de microprograma i el senyal de salt).

Harvard vs. Von Neumann

- L'**Arquitectura Harvard** és una manera de construir les màquines amb una memòria per a les instruccions i un altra per a les dades. És més ràpida que la Von Neumann.

- L'**Arquitectura Von Neumann** construeix la màquina de manera que fa servir una mateixa memòria per a dades e instruccions. Aprofita millor la memòria principal que la Harvard.

YASP

- És un exemple de màquina amb arquitectura Von Neumann
- El camí de dades fa servir els següents registres:
  - Comptador de programa (PC)
  - Registre d'instruccions (IR)
  - Registre temporal de memòria (MBR)
  - Registre d'adreces (MAR)
  - Acumulador (A) – Acumula el resultat de la suma d'una seqüència de bytes
  - Registre auxiliar (X)
- Formes bàsiques d'accedir als operands de memòria
  - **Immediata**: El valor de l'operand s'especifica juntament amb l'operació.
  - **Directa**: L'adreça de l'operand s'indica amb l'operació.
  - **Indirecta**: La posició de memòria en que hi ha l'operand està guardada en l'adreça que acompanya l'operació.
  - **Indexada**: L'adreça de l'operand s'obté sumant al registre índex X a l'adreça que s'adjunta amb l'operació.

3.3.- Processadors

- La **microarquitectura** descriu com s'organitza un processador per implementar un determinat repertori d'instruccions.
- En un **pipeline** passa un flux de processament d'instruccions *en cascada* que es tracta a cada segment de la canonada, de manera que no cal esperar que es processi totalment una porció del flux per tractar-ne un altra diferent.
  - Ho troben tant en RISC com en CISC
- La microarquitectura **multi-core** permet executar diversos programes simultàniament.
- La **CPU** és la màquina algorísmica que interpreta les instruccions d'una ISA determinada. Inclou la unitat de control.
- La **memòria cau** o **caché** proporciona a la CPU un accés més ràpid a la informació de la memòria principal.

Cicle d'execució d'una instrucció

1. Lectura d'instruccio (LI)
2. L'ectura d'argument (LA) – pot ser buida si no hi ha argument
3. Càlcul d'operand (CO) – pot requerir una nova lectura si és directe.
4. Execució de l'operació (EO)

RISC vs. CISC

- CISC
  - Requereixen molts recursos
    - Rang d'operacions molt ampli
    - Fan servir moltes variables
    - Operacions aritmeticològiques amb dades a memòria.
    - Ex. YASP
- RISC
  - Requereixen menys recursos
    - Repertori d'instruccions petit
    - Única instrucció per lectura i única instrucció per emmagatzematge a memòria (load/store)
    - Ex. Femtoproc

Processadors segons propòsit

- Processadors de propòsit general
  - GPP
- Processadors de propòsit específic
  - DSP (Digital signal processors)
  - GPU (Graphics Processing Units)
  - MCU (Microcontroller Unit)

3.4.- Computadors

- L'**arquitectura bàsica d'un computador** s'organitza en perifèrics d'entrada, perifèrics de sortida, perifèrics d'entrada / sortida i processador. (La memòria formaria part del processador o dels perifèrics)
- La **memòria** conté les dades i les instruccions del programa.

- El **mòdul d'entrada/sortida** fa de pont entre el perifèric i el processador, adaptant les diferents maneres de funcionar, el format de les dades i la velocitat del treball.
- Un **bus** connecta dos més components entre ells.
- Els **mòduls d'accés directe a memòria** (DMA) alliberen les CPU de les transferències d'informació entre perifèrics i memòria principal.

Qüestionari

P.- Què és la unitat central de processament o CPU?
R.- La màquina algorísmica que interpreta les instruccions d'una ISA determinada.

P.- Què conté la memòria d'un computador?
R.- Les dades i les instruccions dels programes que pot executar.

P.- Què és una unitat aritmeticològica o ALU?
R.- Un recurs de càlcul programable.

P.- Per a què es fa servir la memòria cau o *caché*?
R.- Per proporcionar a la CPU un accés més ràpid a la informació de la memòria principal.

P.- Quins són els elements en què s'organitza l'arquitectura bàsica d'un computador?
R.- Perifèrics d'entrada, perifèrics de sortida, perifèrics d'entrada/sortida i processador.

P.- Quines formes bàsiques d'accedir als operands en memòria hi ha?
R.- Immediata, directa, indexada i indirecta.

P.- En què consisteix el càlcul de la instrucció següent?
R.- Consisteix en determinar l'adreça de memòria de la instrucció següent.

P.- Per a què es fan servir els mòduls d'accés directe a memòria (DMA)?
R.- Es fan servir per alliberar les CPU de les transferències d'informació entre perifèrics i memòria principal.

P.- Quin tipus de processador permetria executar diversos programes simultàniament?
R.- Un processador amb diverses unitats de processament o *cores*.

P.- En un processador microprogramat, quina funció té el seqüenciador?
R.- Té la funció de controlador d'una màquina d'interpretació d'instruccions.

P.- Quin és el primer pas en el cicle d'execució d'instruccions?
R.- El primer pas és la lectura de la instrucció a executar.

P.- Quin conjunt de registres té el camí de dades d'un processador mínim amb arquitectura de Von Neumann com el *YASP*?
R.- Té els registres següents: Comptador de programa (*PC*), registre d'instruccions (*IR*), registre temporal de memòria (*MBR*), registre d'adreces (*MAR*), acumulador (*A*) i registre auxiliar (*X*).

P.- Què és el llenguatge de màquina?
R.- El llenguatge en què es codifiquen els programes que interpreta la màquina corresponent.

P.- Quina és la funció dels busos en un computador?
R.- Connectar dos o més components entre ells.

P.- Què és una microinstrucció?
R.- Un conjunt d'operacions que es fan en la unitat de processament d'un processador en un cicle de rellotge.

P.- Per a què es fa servir el PC (program counter)?
R.- Per guardar-hi la posició de memòria de la instrucció a executar.

P.- Què descriu la microarquitectura?
R.- Com s'organitza un processador per implementar un determinat repertori d'instruccions.

P.- Que fa una màquina algorísmica general?
R.- Interpreta les instruccions d'un programa emmagatzemat en una memòria

P- Quin és el repertori d'instruccions que interpreta el Femtoproc?
R.- Suma (ADD), negació (NOT), producte lògic (AND) i salt condicional si el resultat de l'última operació ha estat zero (JZ)

P.- Quina arquitectura aprofita millor la memòria i quina pot ser més ràpida?
R.- La de Von Neumann aprofita millor la memòria principal però la de Harvard és més ràpida.

P.- Què és un mòdul d'entrada/sortida?
R.- Un dispositiu que fa de pont entre el processador i els altres components d'un computador.

P.- Per què està constituïda la memòria principal?
R.- Per la RAM.

P.- Quina característica és més exclusiva de les arquitectures de conjunts d'instruccions (ISA) reduïts (RISC) respecte dels complexos (CISC)?
R.- Tenir dos tipus d'instruccions per accedir a dades en memòria: una per a lectura i un altre per a escriptura.

P.- Què és l'arquitectura Von Neumann?
R.- Una manera de construir màquines que tenen una memòria comuna per a les instruccions i les dades.

P.- A part dels processadors de propòsit general, quins altres s'han vist?
R.- DSP, GPU i MCU

P.- Què és l'arquitectura de Hardvard?
R.- Una manera de construir màquines que tenen una memòria per a les instruccions i una per a les dades.