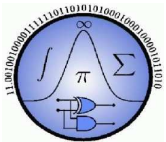


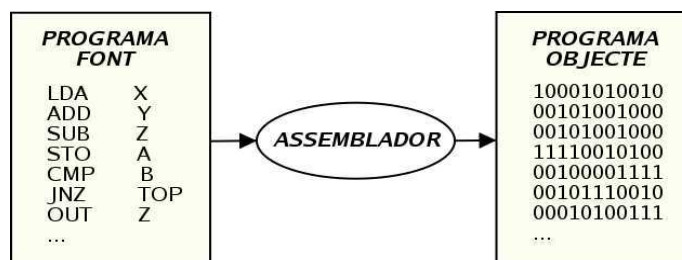
REPRESENTACIÓ DE DADES A MEMÒRIA

- LA MEMÒRIA D'UN COMPUTADOR CONTÉ SEQÜÈNCIES DE BITS
 - AQUESTES SEQÜÈNCIES TENEN SIGNIFICAT:
 - **PROGRAMES**
 - INSTRUCCIONS ASSEMBLADOR CODIFICADES
 - **DADES**
 - CADENES DE CÀRACTERS CODIFICADES (p.e. en **ASCII**)
 - NÚMEROS ENTERS
 - Amb signe, sense signe, **BCD**, (de 8,16, 32 bits, etc...)
 - NÚMEROS REALS
 - Coma fixe, coma flotant (de simple o doble precisió)



PROGRAMES

- ELS MNEMÒNICS ASSEMBLADOR ES TRADUEIXEN A LENGUATGE MÀQUINA

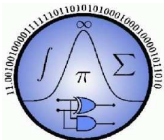


- ELS BITS QUE HI HA A MEMÒRIA SERAN CONSIDERATS INSTRUCCIONS QUAN EL **PC** HI APUNTA PER TAL DE FER UN **FETCH**



CADENES DE CARÀCTERS

- CODIFICACIÓ DELS CARÀCTERS MITJANÇANT NÚMEROS
 - **CODI ASCII** (7/8 bits per caràcter)
 - 128 Caràcters fixes [0..127]
 - '1' → 49_{10}
 - 'A' → 65_{10}
 - 128 Caràcters variables [128..255] (**ASCII Extès**)
 - **UNICODE** (16 bits per caràcter)
 - Cada caràcter és fixe independentment de tot.
 - És bo perquè hi ha idiomes amb més de 255 caràcters (Japonès). **UNICODE** codifica **TOTS** els idiomes de cop.



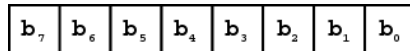
REPRESENTACIÓ NUMÈRICA

- DOS TIPUS BÀSICS DE NÚMEROS
 - **ENTERS**
 - SENSE SIGNE (*Utilitzat*)
 - AMB SIGNE
 - MÒDUL + SIGNE
 - COMPLEMENT A 1
 - COMPLEMENT A 2 (*Utilitzat*)
 - **REALS**
 - COMA FIXE (*Poc utilitzat*)
 - COMA FLOTANT (*Utilitzat*)



ENTERS SENSE SIGNE

- **REPRESENTACIÓ (amb $n = 8$ bits)**



- **Valor** = $b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + \dots + b_{n-1} * 2^{n-1}$

- **Rang**: $[0 .. 2^n - 1]$

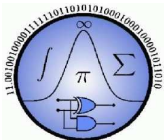
- **Exemples**: $10_d = 0A_h = 00001010_b$ $138_d = 8A_h = 10001010_b$

- **AVANTATGES**

- Tots els bits s'utilitzen pel mòdul

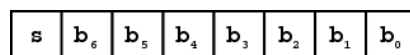
- **INCONVENIENTS**

- No es poden representar valors negatius



ENTERS AMB SIGNE (I): MÒDUL + SIGNE

- **REPRESENTACIÓ (amb $n = 8$ bits)**



- **Valor** = $(-1)^s * (b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + \dots + b_{n-2} * 2^{n-2})$

- **Rang**: $[-2^{n-1} + 1 .. 2^{n-1} - 1]$

- **Exemples**: $+10_d = 0A_h = 00001010_b$ $-10_d = 8A_h = 10001010_b$

- **AVANTATGES**

- Es poden representar valors negatius

- Representació simètrica

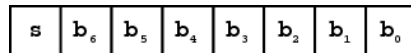
- **INCONVENIENTS**

- Hi ha dues representacions pel 0



ENTERS AMB SIGNE (II): COMPLEMENT A 1

• REPRESENTACIÓ (amb $n = 8$ bits)



- Valor

- Positiu ($S = 0$): $b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + \dots + b_{n-2} * 2^{n-2}$

- Negatiu ($S = 1$): $-((1-b_0) * 2^0 + (1-b_1) * 2^1 + (1-b_2) * 2^2 + \dots + (1-b_{n-2}) * 2^{n-2})$

- Rang: $[-2^{n-1} + 1 .. 2^{n-1} - 1]$

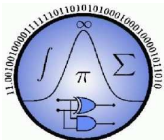
- **Exemples:** $+10_d = 0A_h = 00001010_b$ $-10_d = F5_h = 11110101_b$

• AVANTATGES

- Es poden representar valors negatius
- Representació simètrica

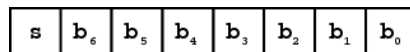
• INCONVENIENTS

- Hi ha dues representacions pel 0



ENTERS AMB SIGNE (III): COMPLEMENT A 2

• REPRESENTACIÓ (amb $n = 8$ bits)



- Valor

- Positiu ($S = 0$): $b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + \dots + b_{n-2} * 2^{n-2} + b_{n-1} * 2^{n-1}$
- Negatiu ($S = 1$): $- (\text{Valor en Complement a 1} + 1)$

- Rang: $[-2^{n-1} .. 2^{n-1} - 1]$

- **Exemples:** $+10_d = 0A_h = 00001010_b$ $-10_d = F6_h = 11110110_b$

• AVANTATGES

- Es poden representar valors negatius
- La suma en *Binari Natural* permet sumar i restar sense signe i amb C_2
- Representació única pel 0

• INCONVENIENTS

- Representació asimètrica



SUMA EN COMPLEMENT A 2

- LA SUMA EN COMPLEMENT A 2 ÉS LA MATEIXA QUE EN BINARI NATURAL (despreçant el *carry*)
- CÀLCUL DE L'OVERFLOW
 - Amb el bit de més pes dels operands i del resultat es pot calcular si hi ha hagut *Overflow*

- Exemples:

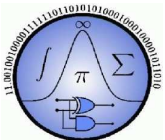
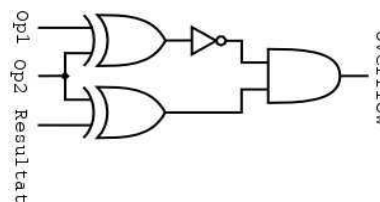
0 000 1010_b 0A_h 10_d
1 111 1101_b FD_h 253_d $C_2(3) = -3$
1 0 000 0111_b 07_h 7_d **Carry!**

0 110 1110_b 6E_h 130_d

0 001 1110_b 1E_h 40_d

1 000 1100_b 9B_h 140_d $C_2(116) = -116$ **Overflow!**

- Circuit per detectar l'Overflow:
(Bit 7 d'Op1, Op2 i Resultat)



REALS (I): COMA FIXA

- S'UTILITZA QUAN:

- No es disposa d'unitat de coma flotant ni per *Hardware* ni *emulada per Software*
- Els càlculs en coma flotant són costosos
- No es necessita gaire precisió o s'utilitza d'un número fix i petit de decimals

- TÈCNICA

- Separar la representació en 3 blocs fixos:
 - El signe
 - La part entera
 - La part decimal



REALS (II): EXEMPLES EN COMA FIXA

- Representació de **N** amb **n** bits = $n_e + n_f + 1$:
 - n_e bits per la part entera
 - n_f bits per la part decimal
 - 1 bit pel signe

Exemple: Representar en coma fixa el número **N = -4,327** amb $n_e = 5$ i $n_f = 6$

El $4_{10} = 100_2$

El $0,327_{10} \rightarrow$

- $0,327 \times 2 = 0,654$
- $0,654 \times 2 = 1,308$
- $0,308 \times 2 = 0,616$
- $0,616 \times 2 = 1,232$
- $0,232 \times 2 = 0,464$
- $0,464 \times 2 = 0,928$

El número en coma fixa és:

1	00100	010100
---	-------	--------

Lògicament té un interval de representació, així com també cal tenir en compte que es comet un error

$$N = - (4 + 0,25 + 0,0625) = -4,3125$$



REALS(III): COMA FLOTANT

- AVANTATGES
 - NECESSITAT DE PODER REPRESENTAR VALORS MOLT GRANS I VALORS MOLT PETITS AL MATEIX TEMPS
 - NO UTILITZAR UN NOMBRE DE BITS EXAGERAT
- INCONVENIENTS
 - PÈRDUA DE PRECISIÓ
 - L'ERROR AUGMENTA COM MÉS LLUNY DE L'INTÈRVAL [-1..1] ES TROBEN ELS VALORS



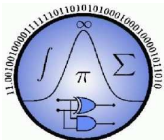
REALS(IV): COMA FLOTANT REPRESENTACIÓ

- Definició de **signe**, **exponent** i **mantissa** en els bits que representen els valors

S	EXPONENT	MANTISSA
---	----------	----------

$$\text{Valor} = (-1)^{\text{Signe}} * \text{Mantissa} * \text{Base}^{\text{Exponent}}$$

- Es pot definir la *base* (2,4,8,10), els bits dedicats a l'exponent i a la mantissa i el format d'aquests
- L'IEEE 754 defineix un estàndard per reals de simple precisió (32 bits) anomenat **floats** i de doble precisió (64 bits) anomenats **doubles**



REALS (V):FLOAT

- FORMAT D'UN **FLOAT** (32 BITS)

b ₃₁	b ₃₀	b ₂₉	b ₂₈	b ₂₇	b ₂₆	b ₂₅	b ₂₄	b ₂₃	b ₂₂	b ₂₁	b ₂₀	b ₁₉	b ₁₈	b ₁₇	b ₁₆	b ₁₅	b ₁₄	b ₁₃	b ₁₂	b ₁₁	b ₁₀	b ₉	b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
S	EXPONENT								s ₁	s ₂	s ₃	MANTISSA															s ₂₂	b ₂₃			

$$\text{Valor} = (-1)^S * (1 + \text{MANTISSA}) * 2^{\text{EXPONENT}}$$

$$\text{Valor} = (-1)^S * (1 + (s_1 * 2^{-1}) + (s_2 * 2^{-2}) + \dots + (s_{23} * 2^{-23})) * 2^E$$

- Normalitzat:** La mantissa té sempre un 1 implícit (24 bits)
- El camp de l'exponent (8 bits) **conté implícitament el signe** de l'exponent
- Utilitza **notació polaritzada** per l'exponent (desplaçada en un *offset*) (aquest *offset* és 127 pels **floats**)

$$\text{Valor} = (-1)^S * (1 + \text{MANTISSA}) * 2^{(\text{EXPONENT} - 127)}$$

- Utilitzant aquesta notació es poden ordenar els valors pel pes dels bits sense haver d'interpretar el valor que hi ha representat



REALS(VI): EXEMPLE FLOAT

- Com es representa el **-0.75** en un **FLOAT**?

$$-0.75_d = -3/4_d = -3/2^2_d$$

$$-11_b / 2^2_d = -0.11_b$$

Notació científica: $-0.11_b * 2^0$

Normalitzar (primer bit enter ha de ser 1): $-1.1_b * 2^{-1}$

- Com que el valor d'un **float** és: $(-1)^s * (1 + \text{Mantissa}) * 2^{E-127}$

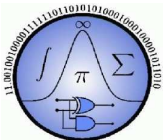
$$(-1)^1 * (1 + .1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 000_b) * 2^{126}$$

- Per tant -0.75_d codificat en un **float** en **coma flotant** és:

$$1\ 0111\ 1110\ 1000\ 0000\ 0000\ 0000\ 0000\ 000_b$$

o també

$$BF40\ 0000_h$$



REALS (VII): DOUBLE

- FORMAT D'UN **DOUBLE** (64 BITS)

b ₃₁	b ₃₀	b ₂₉	b ₂₈	b ₂₇	b ₂₆	b ₂₅	b ₂₄	b ₂₃	b ₂₂	b ₂₁	b ₂₀	b ₁₉	b ₁₈	b ₁₇	b ₁₆	b ₁₅	b ₁₄	b ₁₃	b ₁₂	b ₁₁	b ₁₀	b ₉	b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
S	EXPONENT											s ₁	s ₂	s ₃	MANTISSA											s ₁₉	b ₂₀				

b ₃₁	b ₃₀	b ₂₉	b ₂₈	b ₂₇	b ₂₆	b ₂₅	b ₂₄	b ₂₃	b ₂₂	b ₂₁	b ₂₀	b ₁₉	b ₁₈	b ₁₇	b ₁₆	b ₁₅	b ₁₄	b ₁₃	b ₁₂	b ₁₁	b ₁₀	b ₉	b ₈	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
s ₂₁	s ₂₂	s ₂₃	MANTISSA											s ₃₁	b ₃₂																

- L'**Exponent** té 11 bits i s'utilitza un *offset* a 1023
- El camp **Mantissa** és de 52 bits (amb l'1 implícit 53)

$$\text{Valor} = (-1)^S * (1 + \text{MANTISSA}) * 2^{(\text{EXPONENT} - 1023)}$$



REALS (VII): PROBLEMES

- NO ES PODEN REPRESENTAR TOTS ELS VALORS
- EXISTEIX UN VALOR QUE EXPRESSA LA **MÍNIMA DIFERÈNCIA** QUE HI POT HAVER ENTRE DOS REALS REPRESENTABLES (*Epsilon*)
- ES PRODUEIX ERROR EN ELS CÀLCULS
 - ELS VALORS SÓN APROXIMACIONS
- SI ES TREBALLA NORMALITZAT (ENTRE [-1..1]) ES PRODUEIX MENYS ERROR EN LES OPERACIONS ARITMÈTIQUES