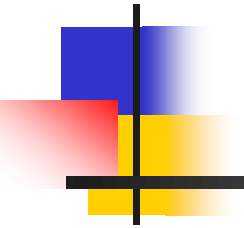
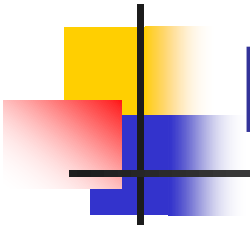


Organización de Computadoras
2021

Turno Recursantes

Clase 1

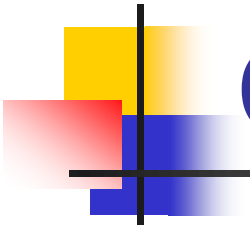


Bibliografía

- ***Organización y Arquitectura de Computadoras – Diseño para optimizar prestaciones***, Stallings W., Editorial Prentice Hall (5ta edición).
- ***Organización de Computadoras***, Tanenbaum A., Editorial Prentice Hall (4ta edición).
- ***Estructura de Computadores y Periféricos***, Martinez Durá R. et al., Editorial Alfaomega, 2001.
- ***Arquitectura de Computadores - Un enfoque cuantitativo***, Hennessy & Patterson., Editorial Mc Graw Hill (1ra edición).

  - http://weblidi.info.unlp.edu.ar/catedras/organiza/

  - Curso "Recursantes Organización de Computadoras" en IDEAS



Conceptos básicos

- Representación de Datos.
  - Números sin signo.
  - Números con signo.
- Operaciones aritméticas
- Banderas de condición
- Otras representaciones:
  - BCH, BCD, Caracteres

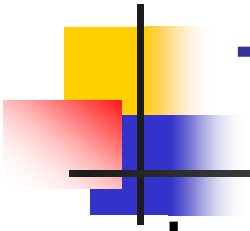
Representación de la información

- Para el manejo de la información, las computadoras utilizan el sistema binario.
- En el sistema binario la mínima unidad de información es el **BIT,** que es una variable que puede tomar 2 valores (niveles eléctricos), a los que típicamente se les asigna (o asocia) 0 y 1.
- De esta manera es más fácil identificar su valor.

Representación de la información (2)

- Es más difícil identificar una variable (eléctrica) de múltiples niveles posibles, que una que solo puede tomar 2.
- Ejemplo :
  - lámpara encendida ó apagada.
  - lámpara encendida con 10 intensidades distintas.

Es más fácil conocer el "estado" de la lámpara en el primer caso (encendida ó apagada), que determinar alguna de las 10 intensidades distintas.



Tipos de datos

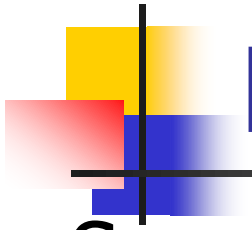
Las computadoras manejan **2 tipos** básicos de datos (binarios):

- Númericos (números)
  - enteros (con/sin signo)
  - reales (con signo)
  - decimales codificados en binario (BCD)
- Alfanuméricos (caracteres)
  - alfabéticos
  - símbolos

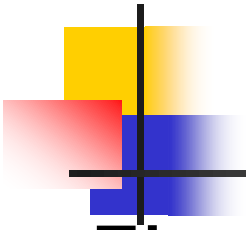
Representación de números enteros

- Sin signo: números positivos
  - Binario sin signo (BSS)
- Con signo: números positivos y negativos
  - Módulo y signo (BCS)
  - Complemento a uno (Ca1) - Complemento a la base reducida
  - Complemento a dos (Ca2) - Complemento a la base
  - Exceso

Números enteros sin signo o Binario sin signo (BSS)



- Se consideran todos los números positivos (incluyendo el 0).
- Cada combinación de bits representa un número BSS distinto. La cantidad de combinaciones distintas depende de la cantidad de bits empleados en la representación y del rango del número a representar.
- La asignación se ordena correlativamente en forma ascendente a partir de 0.



Binario Sin Signo (2)

Ejemplo:

- Quiero representar un número entero positivo en el rango de 0 a 7 (8 números).
- Asignado una combinación por cada número a representar, se necesitan 8 combinaciones.
- Usando 3 bits se tendrán 8 combinaciones.
- Cada combinación se asigna a un número distinto, ordenándolos de forma creciente.

Binario Sin Signo (3)

Resultado:

Representación BSS	Decimal equiv.
000	0
001	1
010	2
.....	...
111	7

8 combinaciones

Binario Sin Signo (4)

Ejemplo: n = 8 bits (combinaciones=256)

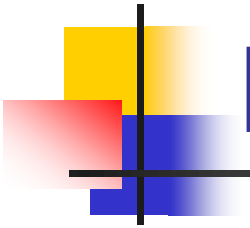
Representación BSS		Decimal
00000000	→	0
.....		.....
01111111	→	127
10000000	→	128
.....		.....
11111111	→	255

256

Binario Sin Signo (5)

- La cantidad de representaciones distintas depende del número n de bits empleado.
- Si se usan n bits, se pueden representar **$\underline{2^n}$** números distintos.
- El rango de un número en BSS es, por lo tanto:

$$0 \longrightarrow (2^n - 1)$$



Números enteros con signo

Existen varias formas de representación de números enteros con signo:

- Binario con Signo BCS (también llamado Módulo y Signo MYS)
- Técnica de Complementos
  - Complemento a 1
  - Complemento a 2
- Técnica de Exceso

Módulo y signo (MYS) o Binario con Signo (BCS)

- 1 bit representa el signo, y los restantes el valor absoluto (o la magnitud).
- Se acostumbra a usar el bit del extremo izquierdo para el signo (bit n-1), y los bits restantes para la magnitud (n-1 bits restantes).
- En n bits:

Binarios Con Signo (2)

- Un 0 en el bit de signo indica que el número es positivo, y un 1 indica que el número es negativo.
- Los n-1 bits restantes representan el valor absoluto en binario.
- Rango de la magnitud: $0 \longrightarrow 2^{n-1} - 1$ (comparar con BSS).
- Rango del número en BCS:

$$-(2^{n-1} - 1) \longrightarrow +(2^{n-1} - 1) \quad \text{con 2 ceros}$$

Binarios Con Signo (3)

- Ejemplos

$+32_{10}$= 00100000 (+)

-32_{10}= 10100000 (−)

32

32

$+7_{10}$=00000111

-7_{10}=10000111

$+41_{10}$=00101001

-41_{10}=10101001

Binarios Con Signo (4)

- Ejemplo con n= 3 bits

000 = +0

001 = +1

010 = +2

011 = +3= $+(2^{n-1} - 1)$

100 = - 0

101 = - 1

110 = - 2

111 = - 3 = $-(2^{n-1} - 1)$

Binarios Con Signo (5)

- Escala numérica n=8 bits

Números positivos:

00000000 ← +0

...

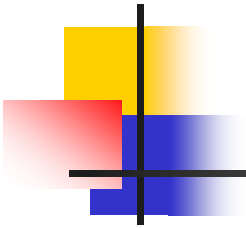
01111111 ← $+(2^{n-1}-1)=+127$

Números negativos:

10000000 ← -0

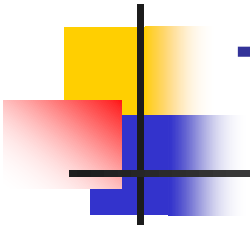
...

11111111 ← $-(2^{n-1}-1)=-127$



Resumen: BCS (6)

- El primer bit sólo indica el signo.
- Los positivos empiezan con cero (0).
- Los negativos empiezan con uno (1).
- Hay dos representaciones del cero.
- El intervalo es simétrico.
- Hay 2^n-1 números distintos (si se considera que el 0 es un solo número).



Técnica de Complementos

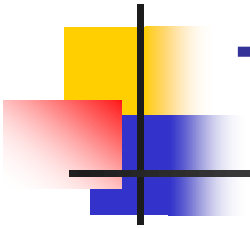
- Definición:

"Se define el complemento de un número A a un número N (con A menor que N) como la cantidad que le falta a A para llegar a N".

Complemento a N de A = N - A

- Propiedad: el complemento a un número N del número (N-A) es igual a A.

Complemento a N de (N-A) = N - (N-A) = A



Técnica de Complementos (2)

Si n es el número de dígitos binarios (bits) a usar para la representación, entonces:

OPCIÓN 1: si N= base^n – 1, es decir N =2^n -1

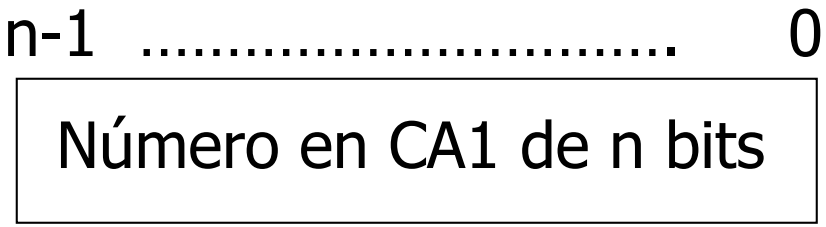
Se llama Complemento a la base disminuída, o: Complemento a 1 (Ca1).

OPCIÓN 2: si N= base^n, es decir N =2^n

Se llama Complemento a la base, o: Complemento a 2 (Ca2)

Representación en Ca1

- Supongamos que se representarán números en Ca1 usando n bits:



Representación en Ca1 (2)

Se define:

- Los números positivos se representan igual que en BSS y BCS.
- Los negativos (o complementarios) se obtienen como Ca1 de los positivos:

N° negativo = Ca1 de N° positivo = N - N° positivo

donde: $N = 2^n - 1$

Representación en Ca1 (3)

- Ejemplo (n=8):

$+7_{10}$= 00000111

entonces:

-7_{10}= N – 00000111 ,es decir

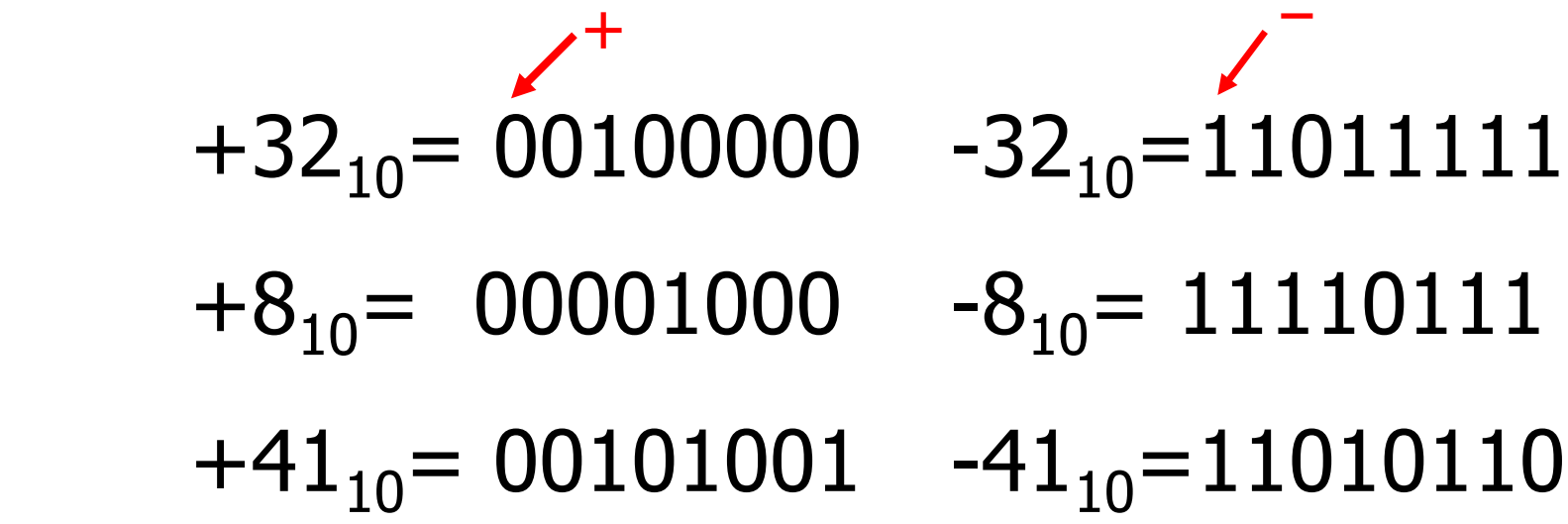
-7_{10}= (2^8 -1) – 00000111 ,o bien

-7_{10}= 11111111 - 00000111

-7_{10}= 11111000

Representación en Ca1 (4)

- Otros ejemplos:



Representación en Ca1 (5)

- Escala numérica con n= 3 bits

000 = +0

001 = +1

010 = +2

011 = +3= $+(2^{n-1} - 1)$

100 = - 3 = $-(2^{n-1} - 1)$

101 = - 2

110 = - 1

111 = - 0

Representación en Ca1 (6)

- Escala numérica con n=8 bits

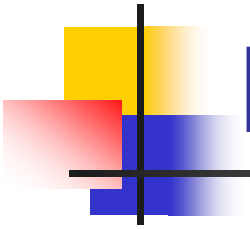
Números positivos	00000000 ←	+0
	...	
	01111111 ←	$+(2^{n-1}-1)=+127$
Números negativos	10000000 ←	$-(2^{n-1}-1)=-127$
	...	
	11111111 ←	-0

Representación en Ca1 (7)

Propiedades:

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- Los números complementarios tienen los bits invertidos
- Dos representaciones del cero (+0 y -0)
- Rango de la representación en CA1:

$$-(2^{n-1}-1) \longrightarrow +(2^{n-1}-1)$$



Representación en Ca1 (8)

Dada una cadena de bits ¿qué número decimal representa si lo interpretamos en Ca1?

- Cuando es positivo (bit mas significativo en 0):

Como siempre:

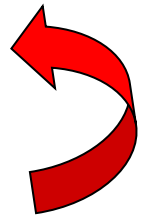
$$01100000 = 1 \times 2^6 + 1 \times 2^5 = 64+32= 96$$

Representación en Ca1 (9)

- Cuando es negativo, puedo hacer dos cosas:
- ✓ Hago el Ca1 del número negativo para obtener el positivo:

Ej.

11100000 = - 31



11100000 → 00011111 = +31

Representación en Ca1 (10)

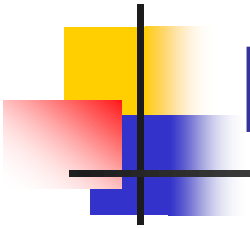
- Otro método:
  - el "peso" del primer dígito ahora es $-(2^{n-1} - 1)$
  - el resto de los dígitos tienen todos peso positivo (como en BSS y BCS).

  Ejemplo:

$11100000 = -1x(2^7 - 1) + 1x2^6 + 1x2^5 =$

$= -127 + 64 + 32 = -31$

- Otro método:
  - Aplicar la definición de Ca1 = $(2^n - 1)$ - N°



Resumen Ca1 (11)

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- Hay dos ceros (uno positivo y otro negativo)
- El intervalo es simétrico
- Hay 2^n-1 números distintos (si se considera que el 0 es un solo número)

Representación en Ca2

- Supongamos que se representarán números en Ca2 usando n bits :

n-1 0

Número en CA2 de n bits

Representación en Ca2 (2)

Se define:

- Los números positivos se representan igual que en BSS y BCS y Ca1
- Los negativos (o complementos) se obtienen como Ca2 de los positivos:

  N° negativo= Ca2 de N° positivo = N - N° positivo

  donde: $N= 2^n$

Representación en Ca2 (3)

- Ejemplo (n=8)

$+32_{10}$= 00100000

entonces:

-32_{10}= N - 00100000

-32_{10}= 2^8 - 00100000

-32_{10}= 100000000 - 00100000

es decir:

Representación en Ca2 (4)

- Hagamos la cuenta en base 2

	1 1 1		
	1	0 0 0 0 0 0 0 0	$= 2^n$
—		0 0 1 0 0 0 0 0	$= 32$
-32 =		1 1 1 0 0 0 0 0	⬅ en Ca2

Representación en Ca2 (5)

- Regla práctica:

  $+32_{10}$= 00100000 ⇐ "mirando" desde la derecha

  -32_{10}= 11100000

- ✓ Los dígitos en rojo se copiaron igual
- ✓ Los dígitos en azul se invirtieron

Representación en Ca2 (6)

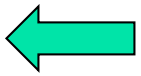
- Otra opción: sumar 1 al Ca1

$+32_{10}$=00100000

1) Ca1 de $+32_{10}$=11011111

2) Sumando 1 al Ca1 de $+32_{10}$

$$\begin{array}{rr} & 11011111 \\ + & \qquad\quad 1 \\ \hline -32_{10}= & 11100000 \end{array}$$

 en Ca2

Representación en Ca2 (7)

- Escala numérica con n= 3 bits

000 = +0

001 = +1

010 = +2

011 = +3= $+(2^{n-1} - 1)$

100 = - 4 = $-(2^{n-1})$

101 = - 3

110 = - 2

111 = - 1

Representación en Ca2 (8)

- Escala numérica con n=8 bits

Números positivos	00000000	+0
	...	
	01111111	$+(2^{n-1}-1)=+127$
Números negativos	10000000	$-(2^{n-1})=-128$
	...	
	11111111	-1

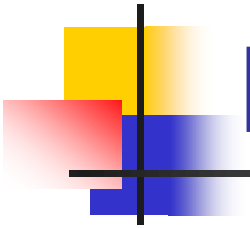
Representación en Ca2 (9)

Propiedades:

- Los positivos empiezan con cero (0).
- Los negativos empiezan con uno (1).
- Una sola representación del cero (porqué?).
- Rango de representación en CA2:

$$-(2^{n-1}) \longrightarrow +(2^{n-1} - 1)$$

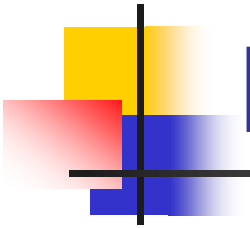
(porqué?)



Representación en Ca2 (10)

Para obtener el Ca2 de un número puedo:

- ✓ Opción 1: aplicar la definición : b^n - N^o
- ✓ Opción 2: calcular el Ca1 y sumarle 1 al resultado.
- ✓ Opción 3: usar regla práctica, "mirando" el número desde la derecha, repetir los dígitos binarios hasta el primer bit en 1 inclusive, y luego invertir los demás bits.



Representación en Ca2 (11)

Dada una cadena de bits ¿qué número decimal representa si lo interpretamos en Ca2?

- Cuando es positivo:

Como siempre

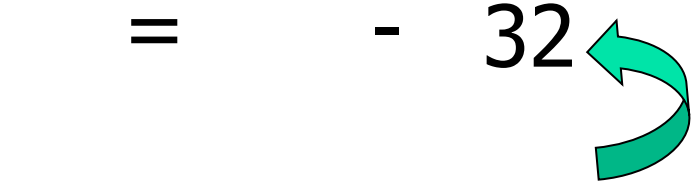
$$01100000 = 1 \times 2^6 + 1 \times 2^5 = 64 + 32 = 96$$

Representación en Ca2 (12)

- Cuando es negativo, puedo hacer:

- ✓ Complementar (en CA2) el número y obtener el positivo. Ejemplo:

11100000 = - 32

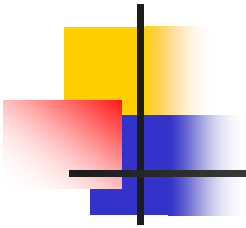


11100000 → 00100000 = +32

Representación en Ca2 (13)

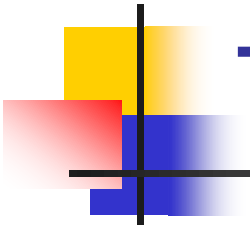
- Otro método: el peso que tiene el bit más significativo ahora es $-(2^{n-1})$ y el resto de los bits con pesos positivos como siempre

$$11100000 = -1x2^7 + 1x2^6 + 1x2^5$$
$$= -128 + 64 + 32 = -32$$



Resumen Ca2 (14)

- Los positivos empiezan con cero (0).
- Los negativos empiezan con uno (1).
- Hay un solo cero.
- Hay un negativo más.
- El rango de números es asimétrico.
- Hay 2^n números distintos.



Técnica de Exceso

- La representación de un número A en exceso E es la que corresponde de sumar al número A el valor constante E (o exceso).

  Exceso E de A = A + E

- Dado un número A representado en exceso E, el número A se obtiene RESTANDO al número en exceso el valor constante E (exceso).

  A = (Exceso E de A) - E

- El signo del número A es el que se obtiene de la resta.

Representación en Exceso 2^{n-1}

Ejemplo:

Supongamos n=6, y E = $2^{n-1} = 2^5 = 32$

Número en E	Número decimal
000000_2	$= 0 - 32 = -32_{10} = -2^{(6-1)}$
.....	
100000_2	$= 32 - 32 = 0_{10}$
.....	
111111_2	$= 63 - 32 = 31_{10} = 2^{(6-1)}-1$

Representación en Exceso 2^{n-1}

- Rango de representación de E 2^{n-1}

$$-2^{(n-1)} \leq x \leq 2^{(n-1)}-1$$

- Una sola representación del 0.
- No sigue la regla de las representaciones anteriores: los positivos empiezan con 1, y los negativos con 0.
- Rango asimétrico.

Sistemas Posicionales

$3574_{10} = 3000 + 500 + 70 + 4$

$= 3 \times 10^3 + 5 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$

- 3 unidades de mil + 5 centenas + 7 decenas + 4 unidades

- Teorema Fundamental de la Numeración

$$N° = \sum_{i=-m}^{n} (dígito)_i \times (base)^i$$

$$\ldots + x_4 \times B^4 + x_3 \times B^3 + x_2 \times B^2 + x_1 \times B^1 + x_0 \times B^0 + x_{-1} \times B^{-1} + x_{-2} \times B^{-2} + \ldots$$

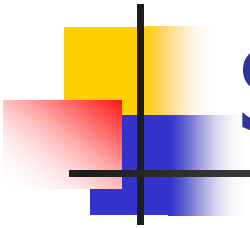
Sistema Decimal

Ejemplo en Base 10

- Dígitos {0,1,2,3,4,5,6,7,8,9}

$3{,}1416_{10} = 3x10^{0} + 1x10^{-1} + 4x10^{-2} + 1x10^{-3} + 6x10^{-4}$

$$\ldots + x_4 \times B^4 + x_3 \times B^3 + x_2 \times B^2 + x_1 \times B^1 + x_0 \times B^0 + x_{-1} \times B^{-1} + x_{-2} \times B^{-2} + \ldots$$



Sistema Binario

Ejemplo en Base 2

- Dígitos {0,1}

$$1001,1_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1}$$

$$= 8 + 0 + 0 + 1 + 0,5$$

$$= 9,5_{10}$$

Sistema Hexadecimal

Ejemplo en Base 16

- Dígitos {0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F}
  10,11,12,13,14,15

$$2CA,8_{16} = 2 \times 16^2 + C \times 16^1 + A \times 16^0 + 8 \times 16^{-1}$$

$$= 512 + 192 + 10 + 0,5$$

$$= 714,5_{10}$$

Números en punto fijo (1)

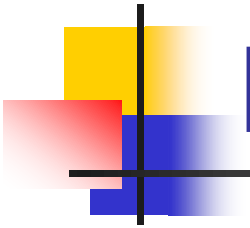
- Se considera que todos los números a representar tienen exactamente la misma cantidad de dígitos y la coma fraccionaria está siempre ubicada en el mismo lugar.
- En sistema decimal sería:

0,23 ó 5,12 ó 9,11

En los ejemplos cada número tiene tres dígitos, y la coma está a la derecha del mas significativo.

Números en punto fijo (2)

- En sistema binario:
  $11{,}10\ (3{,}5)_{10}$ ó $01{,}10\ (1{,}5)_{10}$ ó $00{,}11\ (0{,}75)_{10}$
  Hay 4 dígitos y la coma está entre el 2do y 3er dígito.
- La diferencia principal entre la representación en el papel y su almacenamiento en computadora, es que no se guarda coma alguna, se supone que está en un lugar determinado.



Punto Fijo: Rango y Resolución

Rango: diferencia entre el número mayor y el menor

Resolución: diferencia entre dos números consecutivos

- Por ejemplo, dado un número decimal de 3 dígitos, con la coma después del 1er dígito:

  Rango: es de 0,00 a 9,99 ó [0,00...9,99]

  Resolución: es 0,01

  2,32 - 2,31 = 0,01 o 9,99 - 9,98 = 0,01

Punto Fijo: Rango y Resolución (2)

- Si mantenemos tres dígitos y desplazamos la coma hasta la derecha del último dígito:

  Rango: es de 000 a 999 ó [000...999]

  Resolución: es 1

- Aumenta el rango y disminuye la resolución, es decir hay un compromiso entre rango y resolución.
- En cualquier caso, siempre hay 10^3 números distintos.

Ejemplo en BSS con 4 bits

4 parte ent. y 0 parte frac.

- - - -

Rango: [0…15]

Resolución: 0001 – 0000 =

$0001_2 = 1_{10}$

Binario	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Ejemplo en ... (1)

3 parte ent. y 1 parte frac.

- - - , -

Rango: [0…7,5]

Resolución: 000,1 – 000,0 =

$000,1_2 = 0,5_{10}$

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

Ejemplo en ... (2)

2 parte ent. y 2 parte frac.

- - , - -

Rango: [0…3,75]

Resolución: 00,01 – 00,00 =

$0{,}01_2 = 0{,}25_{10}$

Binario	Decimal
00,00	0
00,01	0,25
00,10	0,5
00,11	0,75
01,00	1
01,01	1,25
01,10	1,5
01,11	1,75
10,00	2
10,01	2,25
10,10	2,5
10,11	2,75
11,00	3
11,01	3,25
11,10	3,5
11,11	3,75

Ejemplo en ... (3)

1 parte ent. y 3 parte frac.

- , - - -

Rango: [0…1,875]

Resolución: 0,001 – 0,000 =

$0{,}001_2 = 0{,}125_{10}$

Binario	Decimal
0,000	0
0,001	0,125
0,010	0,25
0,011	0,375
0,100	0,5
0,101	0,625
0,110	0,75
0,111	0,875
1,000	1
1,001	1,125
1,010	1,25
1,011	1,375
1,100	1,5
1,101	1,625
1,110	1,75
1,111	1,875

Ejemplo en ... (4)

parte ent. y 4 parte frac.

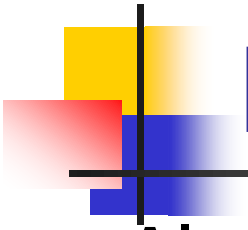
, - - - -

Rango: [0…0,9375]

Resolución:,0001 – ,0000 =

$,0001_2 = 0,0625_{10}$

Binario	Decimal
,0000	0
,0001	0,0625
,0010	0,125
,0011	0,1875
,0100	0,25
,0101	0,3125
,0110	0,375
,0111	0,4375
,1000	0,5
,1001	0,5625
,1010	0,625
,1011	0,6875
,1100	0,75
,1101	0,8125
,1110	0,875
,1111	0,9375

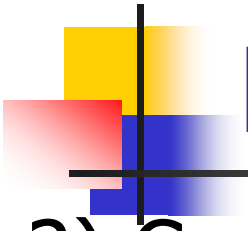


Representación y error

Al convertir un número decimal a binario tendremos 2 situaciones: sin error y con error.

1) Sin error: supongamos que queremos convertir el número decimal 3,125.

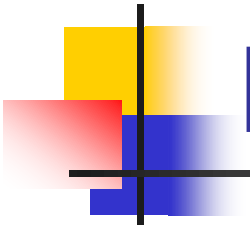
- Sin restricción en la cantidad de bits a usar
  - $3{,}125_{10} = 11{,}001_2$
- Con restricción, por ejemplo 3 bits para parte entera y 4 bits para parte fraccionaria
  - $3{,}125_{10} = 011{,}0010_2$



Representación y error (2)

2) Con error: supongamos que queremos convertir el número decimal 3,2.

- 3 bits para parte fraccionaria:
  - $011{,}001_2 = 3{,}125_{10}$ Error = 3,2 – 3,125 = 0,075
  - $011{,}010_2 = 3{,}250_{10}$ Error = 3,2 – 3,25 = - 0,05
- 4 bits para parte fraccionaria:
  - $011{,}0011_2 = 3{,}1875_{10}$ Error = 3,2 – 3,1875 = 0,0125
- 5 bits para parte fraccionaria:
  - $011{,}00111_2 = 3{,}21875_{10}$ Error =3,2– 3,21875=-0,01875



Representación y error (3)

- El error más pequeño es 0,0125.
- Conclusión: en lugar de representar 3,2 puedo representar 3,1875 usando sólo 4 bits para la parte fraccionaria y cometiendo un error de 0,0125.

Suma y resta en binario

- Una suma en binario se realiza en forma similar a una suma decimal, se suman dígito (binario o bit) a dígito (binario o bit) los n dígitos (bits) de cada número, empezando por los menos significativos (derecha) y considerando los arrastres hacia las posiciones siguientes a la izquierda.
- La resta en binario se realiza en forma similar a una suma decimal. A veces es más sencillo pasar el sustraendo a su complementario y luego sumar.

Suma y resta en binario (2)

- Cuando se suman o restan 2 números es útil detectar ciertas situaciones resultantes de la operación.
- Por ejemplo, si el resultado es cero, o es negativo o positivo.
- Una situación que puede ocurrir también, es que el resultado no entre en la cantidad de bits usados para la representación.

Suma y resta en binario (3)

- Ejemplos en decimal:

1) sumar 2 números de 1 cifra

$$\begin{array}{r} 2 \\ +\ 6 \\ \hline 8 \end{array}$$

resultado entra en 1 cifra

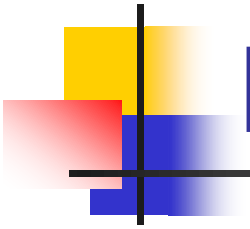
2) sumar 2 números de 1 cifra

$$\begin{array}{r} 5 \\ +\ 6 \\ \hline 11 \end{array}$$

resultado necesita 2 cifras

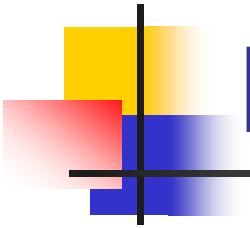
Bits de condición (banderas)

- Para atender las situaciones antes mencionadas el procesador usa unos bits internos denominados BITS DE CONDICIÓN (o flags) que reflejan el resultado de una operación aritmética previa.
- Sus valores ("estados") permiten tomar decisiones tales como decidir si se toma un camino u otro (transferencia de control), o relaciones entre números (mayor, menor, igual).



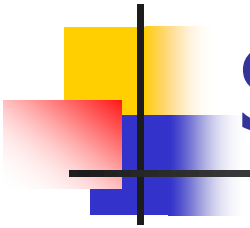
Banderas aritméticas

- Z (cero): vale 1 si el resultado de la operación son todos bits 0.
- C (carry): en la suma vale 1 si hay acarreo del bit más significativo; en la resta vale 1 si hay 'borrow' hacia el bit más significativo.
  - Cuando la operación involucra números sin signo, C=1 indica una condición fuera de rango.



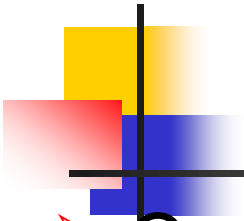
Banderas aritméticas

- N (negativo): igual al bit más significativo del resultado.
  - Es 1 si el resultado es negativo
- V (overflow): en 1 indica una condición de fuera de rango (desborde) en Ca2.
  - El resultado no se puede expresar con el número de bits utilizado.



Suma en Ca2 y desborde

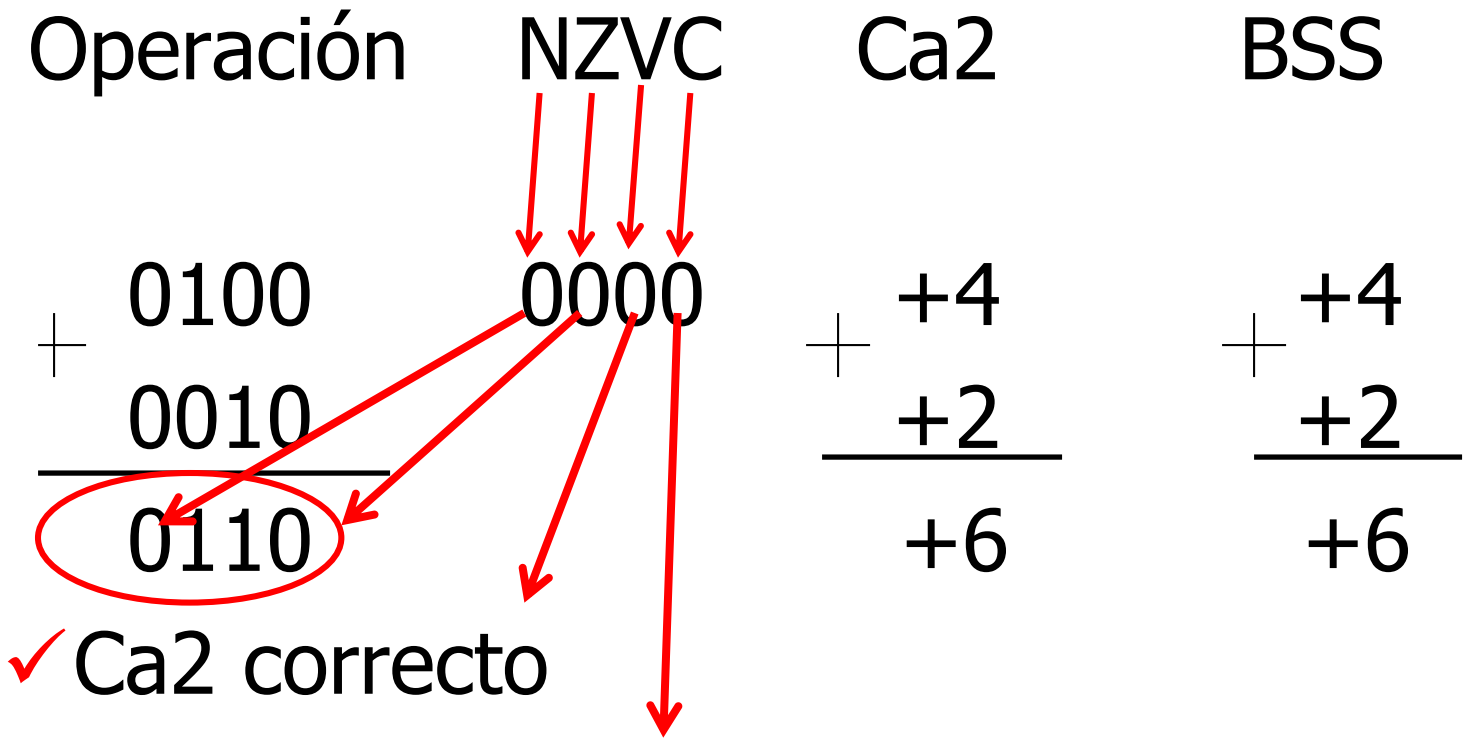
- Overflow (desborde): ocurre cuando al sumar 2 números, el resultado no entra en la cantidad de bits usados.
- Cuando hay Overflow en la suma?
  - Sumando dos números + y el resultado es –
  - Sumando dos números – y el resultado es +
- Cuando NO hay Overflow en la suma?
  - Sumando números de distinto signo.



Resta en Ca2 y desborde

- Overflow (desborde): ocurre cuando al restar 2 números, el resultado no entra en la cantidad de bits usados.
- Cuando hay Overflow en la resta?
  - Restar a un número + un – y el resultado es -
  - Restar a un número – un + y el resultado es +
- Cuando NO hay Overflow en la resta?
  - Restando números de igual signo.

Bits de condición para la suma



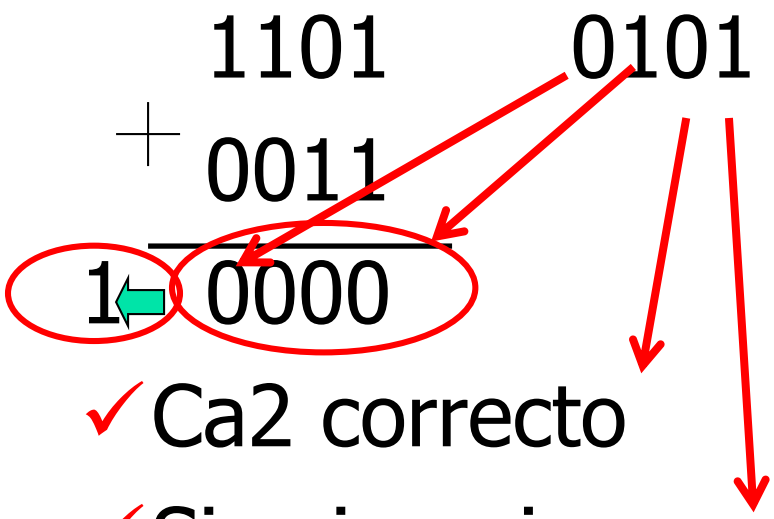
Bits de condición para la suma

Operación	NZVC	Ca2	BSS
0101 + 0111 = 1100	1010	+5 + +7 = V -4	+5 + +7 = +12

- ✓ Ca2 incorrecto
- ✓ Sin signo correcto

Bits de condición para la suma

Operación	NZVC	Ca2	BSS
1101	0101	-3	13
+ 0011		+ +3	+ 3
1 0000		0	C 0



- ✓ Ca2 correcto
- ✓ Sin signo incorrecto

Bits de condición para la suma

Operación	NZVC	Ca2	BSS
1001 + 1100 = 0101	0011	-7 + -4 = +5	9 + 12 = 5
		V	C

- ✓ Ca2 incorrecto
- ✓ Sin signo incorrecto

Bits de condición para la resta

Operación	NZVC	Ca2	BSS
1001	0010	-7	9
− 0100		− +4	− 4
0101		V +5	5

- ✓ Ca2 incorrecto
- ✓ Sin signo correcto

Bits de condición para la resta

Operación	NZVC	Ca2	BSS
1 ⇨ 0101	1001	+5	5
− 0111		− +7	− 7
1110		-2	B 14

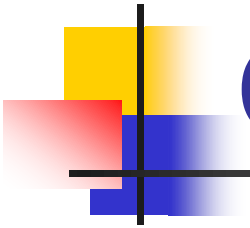
✓ Ca2 correcto

✓ Sin signo incorrecto

Suma en BCS

$$\begin{array}{r} 1\ 001 \\ +\ 1\ 001 \\ \hline 1\ 010 \end{array} \qquad \begin{array}{r} -1 \\ +\ -1 \\ \hline -2 \end{array}$$

- Pensar el algoritmo de suma.



Otras representaciones

- BCH
- BCD
- Caracteres

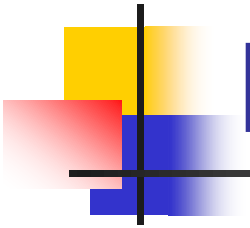
Sistema hexadecimal codificado en binario (BCH)

- Los dígitos hexadecimales se convierten uno a uno en binario
- Para representar un dígito hexadecimal se utilizará siempre 4 bits
- Se asocia cada dígito con su valor en binario puro

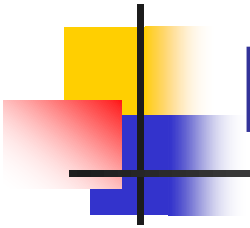
BCH

Dígito hexadecimal	Código BCH
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Sistema decimal codificado en binario (BCD)

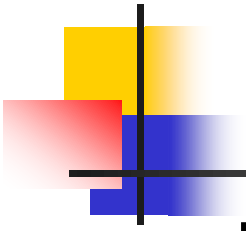


- Los dígitos decimales se convierten uno a uno en binario
- Para representar un dígito decimal se requerirán 4 bits
- Se asocia cada dígito con su valor en binario puro



BCD

Dígito decimal	Código BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

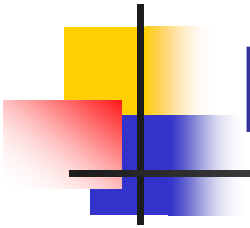


BCD

- Notar que hay 6 representaciones binarias que no se usan en la representación BCD:

1010 (A)
1011 (B)
1100 (C)
1101 (D)
1110 (E)
1111 (F)

Eso significa que si aparece un número binario mayor a 9, no es BCD.



BCD

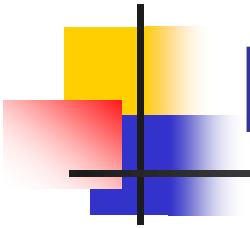
- BCD tiene dos ámbitos de aplicación:

➢ Operaciones de E/S y comunicaciones con periféricos

Los números se codifican usando 8 bits por dígito (un byte) : *BCD desempaquetado.*

➢ En cálculos aritméticos

Los números se codifican usando 4 bits por dígito: *BCD empaquetado.*



BCD Desempaquetado sin signo

- Por cada dígito se usan 8 bits
  - Los 4 bits menos significativos representan el número decimal (dígitos de 0 a 9).
  - Los 4 bits más significativos se completan con "1".
  - Ejemplo

834 = 11111000 11110011 11110100

= F8 F3 F4

BCD Desempaquetado con signo

- Por cada dígito se usan 8 bits
- Los 4 bits menos significativos representan el número decimal (dígitos 0 al 9)
- Los 4 bits más significativos del último dígito representan el signo:
  - C_{16}= 1100 representa al signo +
  - D_{16} = 1101 representa al signo -

BCD Desempaquetado con signo

- Ejemplo:

+ 834 = 11111000 11110011 11000100

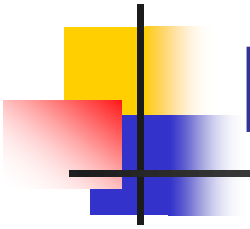
= F8 F3 C4

Los 4 bits que acompañan al último dígito indican el signo.

BCD Desempaquetado con signo

Ejemplo:

- -834 = 11111000 11110011 11010100
  = F8 F3 D4

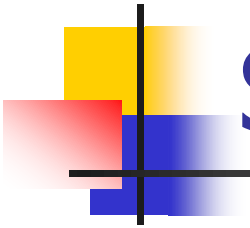


BCD Empaquetado con signo

- Por cada dígito decimal se usan 4 bits.
- Si faltan bits para completar el byte se rellena con 0 en la posición más significativa.

Ejemplo:

- + 834 = 10000011 01001100 = 83 4C
- - 34 = 00000011 01001101 = 03 4D



Suma en BCD

- De las 16 representaciones posibles con 4 bits, usamos 10 para los dígitos 0 al 9
- Nos sobran 6 combinaciones de 4 bits
- Al sumar dos dígitos BCD, se nos presentan dos casos :
  - la suma es ≤ 9
  - la suma es > 9

Suma en BCD

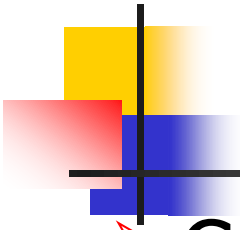
- Suma ≤ 9

decimal		BCD	
	41	0100	0001
+	22	0010	0010
	63	0110	0011

Suma en BCD

- Suma > 9

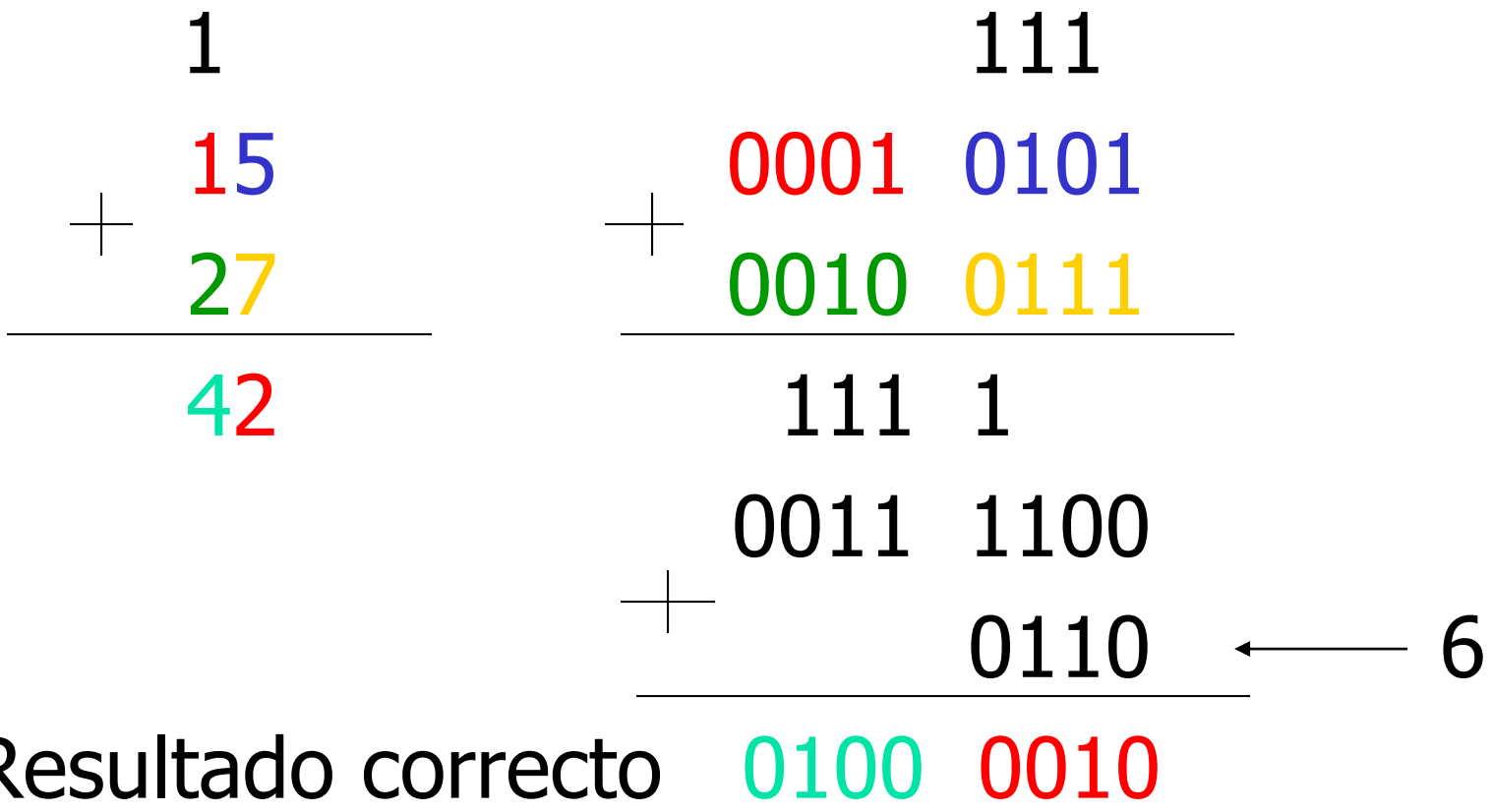
decimal	BCD
1	111
15	0001 0101
+ 27	+ 0010 0111
42	0011 1100
	3 no válido



Suma en BCD

- Cuando la suma de los dos dígitos da >9 podemos compensar (ajustar) el resultado para tener en cuenta la "distancia" entre BCH y BCD.
- Eso se consigue generando el "acarreo" originado por las 6 combinaciones no usadas.
- Conclusión: cuando la suma de los dígitos es > 9 hay que sumar 6 en ese dígito.

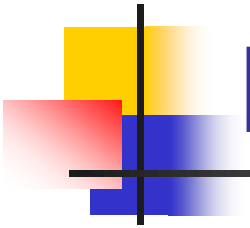
Suma en BCD



Suma en BCD

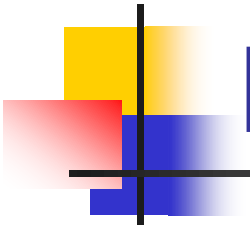
- Ejemplo

```
   1                    111   1
  476              0100 0111 0110
+  55            +       0101 0101
-----            ---------------
 531              0100 1100 1011
                +       0110 0110
                ---------------
                0101 0011 0001
```



Representación alfanumérica

- Letras (mayúsculas y minúsculas)
- Dígitos decimales (0, ..., 9)
- Signos de puntuación
- Caracteres especiales
- “Caracteres” u órdenes de control



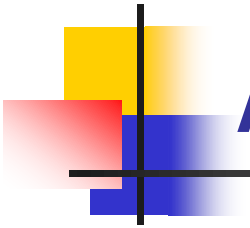
Ejemplo

A cada símbolo un código en binario

Ejemplo: x, y, α, β, #, @, [,]

- Ocho símbolos ¿Cuántos bits? ¿Por qué?

000	x	@	...
001	y	[	
010	α	α	
011	β	#	
100	#	β	
101	@	y	
110	[	x	
111	]	]	

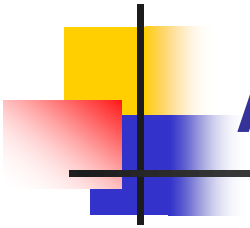


Algunos códigos

- FIELDATA
  - 26 letras mayúsculas + 10 dígitos + 28 caracteres especiales
  - Total 64 combinaciones $\Rightarrow$ Código de 6 bits
- ASCII

American Standard Code for Information Interchange

  - FIELDATA + minúsculas + ctrl
  - Total 128 combinaciones $\Rightarrow$ Código de 7 bits



Algunos códigos (2)

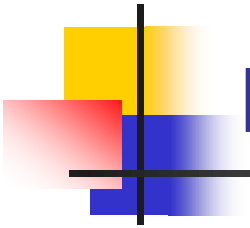
- ASCII extendido
  - ASCII + multinacional + semigráficos + matemática
  - Código de 8 bits
- EBCDIC - Extended BCD Interchange Code
  - similar al ASCII pero de IBM
  - Código de 8 bits

Tabla ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Una extensión al ASCII

128	Ç	144	É	160	á	176	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	⌐	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	⁻
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	₧	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		



mayor información …

- Capítulo 8: Aritmética del computador (8.1., 8.2., 8.3.)
  - Stallings, 5ta Ed.
- Apéndice 8A: Sistemas de Numeración
  - Stallings, 5ta Ed.
- Sistemas enteros y Punto fijo
  - Apunte 1 de Cátedra
- Capítulo 3: Lógica digital y representación numérica
  - Apuntes COC - Ingreso 2013