

Organización de Computadoras

CURSO 2024

TURNO RECURSANTES
CLASE 1

Cronograma Recursantes 2023

Teoría	Viernes	Contenidos/ActividadesTeoría - Aula 5, 18hs
1	15 MARZO	CLASE 1: Representación de la información en Sistemas de Cómputo
2	22 MARZO	CLASE 1: Números con signo. Operaciones aritméticas
	29 MARZO	FERIADO
3	5 ABRIL	CLASE 2: Representación de números en punto flotante
4	12 ABRIL	CLASE 3: Lógica digital. Algebra de Boole. Compuertas lógicas. Circuitos combinatorios
5	19 ABRIL	CLASE 3: Circuitos secuenciales
6	26 ABRIL	CLASE 4: Arquitectura von Neumann
	03 MAYO	Evaluación Teórica Corta 1 (sin dictado de clase)
7	10 MAYO	CLASE 5: Ciclo de instrucción
8	17 MAYO	CLASE 6: Formato de instrucción y modos de direccionamiento
9	24 MAYO	CLASE 7: Registros – Repertorio de instrucciones
10	31 MAYO	CLASE 8: Organización de la memoria – Memoria principal
11	07 JUNIO	CLASE 9: Memoria caché y memoria secundaria
12	14 JUNIO	CLASE 10: Periféricos
	21 JUNIO	FERIADO
	28 JUNIO	Evaluación Teórica Corta 2 (sin dictado de clase)

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***, Martínez 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

Resumen clase 1

- Representación de la información
- Representación de tipos de datos numéricos
 - Números sin signo
 - Números con signo
 - MYS, CA1, CA2, Exc
- Operaciones aritméticas básicas
- Banderas de condición
- Otros tipos de representaciones
 - BCH
 - BCD
 - Caracteres

Representación de la información

5

- Para operar, las computadoras utilizan señales eléctricas.
- Si bien las señales eléctricas pueden adoptar infinitos valores, desde los comienzos de la computación se observó que era más fácil operar con ellas si se asumía que podían tomar solo 2 valores discretos: alto y bajo, en lugar de n valores (por ejemplo 10).
- El valor alto se lo puede asociar al valor 1, y el bajo al 0 (podría ser al revés, también).
- De esta manera se cambia el enfoque de análisis. Se pasa de un problema eléctrico/electrónico a un problema numérico/matemático.

Representación de la información

- El problema de trabajar con múltiples niveles discretos eléctricos (10) o solo 2, tiene que ver con que es más difícil identificar una variable eléctrica con muchos niveles que una con solo 2 niveles.
 - Ejemplo :
 - lámpara encendida o apagada (2 posibles niveles o estados)
 - lámpara encendida con 10 intensidades distintas (10 posibles niveles o estados)
- En el ejemplo anterior, es más fácil conocer el “estado” de la lámpara en el primer caso (encendida o apagada), que determinar alguna de las 10 intensidades distintas.

Representación de la información

7

- Las señales eléctricas se asumen que pueden estar en 1 de 2 posibles estados: alto o bajo. El alto se asocia al valor numérico “1” y el bajo al “0”.
- Una única señal (que puede estar en 0 o en 1) es la mínima unidad de información. Es la variable binaria, porque puede tomar solo 2 posibles valores: 0 o 1.
- Se la conoce como bit.
- Las variables que pueden tomar más de 2 valores usan combinaciones de bits, los suficientes para cubrir todos los posibles valores que puede tener la variable.

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

Los tipos de datos numéricos representan un porcentaje muy alto del tipo de información que manipulan los sistemas de cómputo, y por ello vamos a analizar en detalle sus características.

Representación de números enteros

El primer tipo de dato a analizar son los números enteros.

Pueden ser de 2 tipos:

- 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 - Binario sin signo (BSS)

Los números enteros sin signo, también llamados Binarios Sin Signo (BSS), son combinaciones de 1 y 0 que tienen las siguientes características:

- Todos los números se consideran positivos (incluyendo el 0).
- Se asigna una combinación de bits diferente por cada representación (por cada valor de la variable).
- La cantidad de combinaciones distintas depende de la cantidad de bits empleados en la representación (2^n)
- La cantidad de combinaciones debe abarcar el rango del número a representar.
- La asignación se ordena correlativamente en forma ascendente a partir de 0.

Números enteros sin signo - Binario sin signo (BSS)

Ejemplo:

- Para representar un número entero positivo en el rango de 0 a 7 (8 números), se debe:
 - Asignar una combinación de bits por cada número a representar. Como se quiere representar un número entre 0 y 7 se necesitan 8 combinaciones de bits distintas.
 - Usando 3 bits se tendrán 8 combinaciones.
 - Cada combinación se asigna a un número distinto, ordenándolos de forma creciente a partir del 0.

Números enteros sin signo - Binario sin signo (BSS)

Resultado:

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

8 valores

8 combinaciones

Números enteros sin signo - Binario sin signo (BSS)

Ejemplo: $n = 8$ bits (combinaciones=256)

Representación BSS

Decimal

00000000

0

.....

.....

01111111

127

10000000

128

.....

.....

11111111

255

256 valores

256 combinaciones

Números enteros sin signo - Binario sin signo (BSS)

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

Rango en BSS:

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

Números enteros con signo

Existen varias formas de representación de números enteros con signo:

- Binario con Signo BCS (también llamado Módulo y Signo MYS)
- Complementos, hay 2 variantes:
 - Complemento a 1
 - Complemento a 2
- Exceso

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

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

$n-1$ 0

Número en MYS de n bits

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

Se define la representación en MYS de la siguiente manera.

- El bit más significativo representa únicamente el signo
- Los $n-1$ bits restantes representan el valor absoluto (o la “magnitud”).
- Se acostumbra a usar el bit del extremo izquierdo para el signo (bit de posición $n-1$), y los bits restantes para la magnitud (posiciones 0 a $n-2$).
- En n bits:



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

- 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: $0 \longrightarrow 2^{n-1} - 1$
- Rango del número en BCS:

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

con 2 representaciones del cero

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

Ejemplos:

$$+32_{10} = 00100000_{\text{MYS}}$$

$$-32_{10} = 10100000_{\text{MYS}}$$

$$+7_{10} = 00000111_{\text{MYS}}$$

$$-7_{10} = 10000111_{\text{MYS}}$$

$$+41_{10} = 00101001_{\text{MYS}}$$

$$-41_{10} = 10101001_{\text{MYS}}$$

Se usó un subíndice al final del número para indicar la base numérica empleada en el número. Este subíndice no siempre se usa, en este caso permite identificar la base numérica usada para representarlo.

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

Escala numérica con $n = 3$ bits

$$000_{\text{MYS}} = +0_{10}$$

$$001_{\text{MYS}} = +1_{10}$$

$$010_{\text{MYS}} = +2_{10}$$

$$011_{\text{MYS}} = +3_{10} = +(2^{n-1} - 1)$$

$$100_{\text{MYS}} = -0_{10}$$

$$101_{\text{MYS}} = -1_{10}$$

$$110_{\text{MYS}} = -2_{10}$$

$$111_{\text{MYS}} = -3_{10} = -(2^{n-1} - 1)$$

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

Escala numérica con $n=8$ bits

Números positivos	{	00000000	MYS	$+0_{10}$	
		...			
		01111111	MYS	$+127_{10}$	$+(2^{n-1} - 1)$
Números negativos	{	10000000	MYS	-0_{10}	
		...			
		11111111	MYS	-127_{10}	$-(2^{n-1} - 1)$

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

Propiedades del BCS:

- El primer bit sólo indica el signo.
- Los positivos empiezan con cero (0).
- Los negativos empiezan con uno (1).
- Hay 2 representaciones del cero.
- El intervalo es simétrico.
- Dado que el 0 está duplicado (positivo y negativo) hay en total $2^n - 1$ números distintos, es decir 1 menos que la cantidad en binario sin signo.

Técnicas de Complementos

23

➤ 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 ”.

Es decir:

$$\text{Complemento a } N \text{ de } A = N - A$$

➤ Propiedad: el complemento a un número N del número $(N - A)$ es igual a A , ya que:

$$\text{Complemento a } N \text{ de } (N - A) = N - (N - A) = A$$

Técnicas de Complementos

Las características de la representación en Complementos dependen de la elección del número N . Para el caso de la representación en binario, se pueden usar 2 valores de N :

➤ Si n es el número de bits a usar para la representación, entonces:

➤ OPCIÓN 1: $N = \text{base}^n - 1$, es decir $N = 2^n - 1$

Se llama Complemento a 1 (CA1) o Complemento a la base disminuída.

➤ OPCIÓN 2: $N = \text{base}^n$, es decir $N = 2^n$

Se llama Complemento a 2 (CA2) o Complemento a la base.

Complemento a 1

- Supongamos que se representarán números en CA1 usando n bits:

$n-1$ 0

Número en CA1 de n bits

Complemento a 1

Se define la representación en CA1 de la siguiente manera:

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

Es decir:

$$\begin{array}{l} \text{N}^\circ \text{ negativo} \\ \text{positivo} \end{array} = \text{Ca1 de N}^\circ \text{ positivo} = N - \text{N}^\circ \text{ positivo}$$

donde:

$$N = 2^n - 1$$

Complemento a 1

Ejemplo: supongamos trabajar números en CA1 en 8 bits, es decir $n=8$.

➤ El número 7 positivo se representará como:

$$+7_{10} = 00000111_{CA1}$$

➤ De acuerdo a la definición de CA1, el 7 negativo se obtiene como CA1 del número 7 positivo. Si $N = 2^8 - 1$, entonces:

$$-7_{10} = N - 00000111_{CA1}, \text{ es decir}$$

$$-7_{10} = (2^8 - 1) - 00000111_{CA1}, \text{ o bien}$$

$$-7_{10} = 11111111_{CA1} - 00000111_{CA1}$$

$$-7_{10} = 11111000_{CA1}$$

Complemento a 1

Otros ejemplos:

$$+32_{10} = 00100000_{CA1}$$

$$+8_{10} = 00001000_{CA1}$$

$$+41_{10} = 00101001_{CA1}$$

$$-32_{10} = 11011111_2$$

$$-8_{10} = 11110111_2$$

$$-41_{10} = 11010110_2$$

Complemento a 1

➤ Escala numérica con $n=3$ bits

$$000_{CA1} = +0_{10}$$

$$001_{CA1} = +1_{10}$$

$$010_{CA1} = +2_{10}$$

$$011_{CA1} = +3_{10} = +(2^{n-1} - 1)$$

$$100_{CA1} = -3_{10} = -(2^{n-1} - 1)$$

$$101_{CA1} = -2_{10}$$

$$110_{CA1} = -1_{10}$$

$$111_{CA1} = -0_{10}$$

Complemento a 1

➤ Escala numérica con $n=8$ bits

Números
positivos

00000000	$_{CA1}$	$+0_{10}$	
...		...	
01111111	$_{CA1}$	$+127_{10}$	$+(2^{n-1} - 1)$

Números
negativos

10000000	$_{CA1}$	-127_{10}	$-(2^{n-1} - 1)$
...		...	
11111111	$_{CA1}$	-0_{10}	

Complemento a 1

Propiedades del CA1:

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- Los números complementarios tienen los bits invertidos
- Hay 2 representaciones del cero (+0 y -0)
- Rango de la representación en CA1:

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

Complemento a 1

Dada una cadena de bits en representación CA1 ¿qué número decimal representa?

Cuando es positivo (bit más significativo en 0):

como siempre (es decir, como en BCS)

Ejemplo:


$$01100000 = 1 \times 2^6 + 1 \times 2^5 = 64 + 32 = 96_{10}$$

Complemento a 1

Cuando es negativo, se pueden usar 3 métodos:

➤ Opción 1: hacer el CA1 del número negativo para obtener el positivo:

Ejemplo: qué número decimal representa el número 11100000 representado en CA1?

1) El CA1 del 11100000_{CA1} es  00011111_2

2) El número 00011111_{CA1} es igual $+31_{10}$

3) Entonces 11100000_{CA1} es igual a -31_{10}

Complemento a 1

- Opción 2: por cálculo del “peso” de cada dígito binario, el bit más significativo tiene un peso de:

$$\text{peso del primer dígito} = -(2^{n-1} - 1)$$

el resto de los dígitos tienen todos peso positivo (como en BSS y BCS).

Ejemplo:

$$\begin{aligned} 11100000_{\text{CA1}} &= -1 \times (2^7 - 1) + 1 \times 2^6 + 1 \times 2^5 = \\ &= -127_{10} + 64_{10} + 32_{10} = -31_{10} \end{aligned}$$

- Opción 3: encontrar el complementario positivo, aplicando la definición de $\text{Ca1} = (2^n - 1) - N^\circ$

Complemento a 1

Resumen propiedades del CA1

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- Hay dos ceros (uno positivo y otro negativo)
- El intervalo es simétrico
- Dado que el 0 está duplicado (positivo y negativo) hay en total $2^n - 1$ números distintos, es decir 1 menos que la cantidad en binario sin signo.

Complemento a 2

36

- Supongamos que se representarán números en CA2 usando n bits :

$n-1$ 0

Número en CA2 de n bits

Complemento a 2

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

es decir:

$$\begin{array}{l} \text{N}^\circ \text{ negativo} = \text{Ca2 de N}^\circ \text{ positivo} = N - \text{N}^\circ \\ \text{positivo} \end{array}$$

donde:

$$N = 2^n$$

Complemento a 2

Ejemplo: supongamos trabajar números en CA2 en 8 bits. Es decir $n=8$.

➤ El número 7 positivo se representará como:

$$+7_{10} = 00000111_{CA2}$$

➤ De acuerdo a la definición de CA2, el 7 negativo se obtiene como CA2 del número 7 positivo. Si $N = 2^8$, entonces:

$$-7_{10} = N - 00000111_{CA2}, \text{ es decir}$$

$$-7_{10} = (2^8) - 00000111_{CA2}, \text{ o bien}$$

$$-7_{10} = 100000000 - 00000111_{CA2}$$

haciendo la cuenta:

Complemento a 2

- Hay que restar 0000111 de 10000000:

$$\begin{array}{r}
 1111111 \\
 \hline
 10000000 = 2^n \\
 - \quad 0000111 = +7 \\
 \hline
 -7 = 11111001_{CA2} \quad \text{en Ca2}
 \end{array}$$

- Comparar este resultado con el obtenido para la representación del -7 en CA1 visto anteriormente:

En CA1:

$$-7_{10} = 11111000_{CA1}$$

Complemento a 2

➤ Escala numérica con $n=3$ bits

$$000_{CA2} = +0_{10}$$

$$001_{CA2} = +1_{10}$$

$$010_{CA2} = +2_{10}$$

$$011_{CA2} = +3_{10} = +(2^{n-1} - 1)$$

$$100_{CA2} = -4_{10} = -(2^{n-1})$$

$$101_{CA2} = -3_{10}$$

$$110_{CA2} = -2_{10}$$

$$111_{CA2} = -1_{10}$$

Complemento a 2

➤ Escala numérica con $n=8$ bits

Números positivos	{	00000000	$_{CA2}$	$+0_{10}$	
		
		01111111	$_{CA2}$	$+127_{10}$	$+(2^{n-1} - 1)$
Números negativos	{	10000000	$_{CA2}$	-128_{10}	$-(2^{n-1})$
		
		11111111	$_{CA2}$	-1_{10}	

Complemento a 2

Propiedades del CA2:

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- Hay 1 sola representación del cero
- Dado que hay 1 sola representación del 0, hay en total 2^n números distintos.
- Al tener la misma cantidad de números positivos, y 1 sola representación del 0, se agrega un número negativo más.
- Rango de la representación en CA1:

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

Complemento a 2

Dada una cadena de bits en representación CA2 ¿qué número decimal representa?

Cuando es positivo (bit mas significativo en 0):

como siempre (es decir, como en BCS)

Ejemplo:

$$01100000_{CA2} = 1 \times 2^6 + 1 \times 2^5 = 64 + 32 = 96_{10}$$

Complemento a 2

Cuando es negativo (bit más significativo en 1) tengo varias opciones:

- 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.
- Opción 4: considerando el “peso” que tiene el bit de signo.


Complemento a 2

- Opción 1: aplicar la definición de CA2 para obtener el complementario positivo. Es decir, si N_o es el número negativo que quiero calcular, se debe hacer:

$$2^n - N_o \quad (\text{para obtener el complementario positivo})$$

- Opción 2: calcular el CA1 y sumarle 1

Ejemplo: qué número decimal representa el número 11100000_{CA2} representado en CA2?

- 1) El CA1 del 11100000 es  $0011111 = +31_{10}$
- 2) El número en CA2 es igual al CA1+1, es decir $+32_{10}$
- 3) Entonces 11100000_{CA2} en CA2 es igual a -32_{10}

Complemento a 2

- Opción 3: Usando la siguiente regla práctica, para encontrar el complementario positivo:
 - Se toma el número binario negativo.
 - Se barre el número de derecha a izquierda, empezando por el menos significativo
 - Si el bit es 0, el complementario también es 0.
 - Se repite el barrido hasta encontrar el primer bit en 1. El complementario tendrá ese bit en 1.
 - A partir del primer 1 (excluido) se invierten los bits restantes.

Ejemplo: número negativo: 11100000 barrido ←

Complementario positivo: 00100000 que es el número $+32_{10}$

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

Complemento a 2

- Opción 4: el “peso” que tiene el bit más significativo ahora es $-(2^{n-1})$ y el resto de los bits con “pesos” positivos (como siempre):

Ejemplo:

$$\begin{aligned} 11100000_{\text{CA2}} &= -1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 \\ &= -128_{10} + 64_{10} + 32_{10} = -32_{10} \end{aligned}$$

Complemento a 2

Resumen propiedades del CA2:

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

Técnica de Exceso

- La representación de un número A en exceso E es la que obtiene sumando al número A un valor constante, denominado Exceso E .

$$\text{Exceso } E \text{ de } A = A + E$$

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

$$A = (\text{Exceso } E \text{ de } A) - E$$

- El signo del número A es el que se obtiene de la resta.
- El valor de E es una constante arbitraria, es decir se puede elegir cualquier valor. Pero en la práctica se usa un valor relacionado con la cantidad de bits usados en la representación.

Técnica de Exceso

- Los valores más comunes de E son 2^{n-1} y $2^{n-1} - 1$.
- Es decir, si $n=8$
 - $E = 128$
 - $E = 127$
- Con estos valores, el rango de la representación es (casi) simétrico.

Representación en Exceso $2^n - 1$

Ejemplo 1:

Supongamos que:

$$n=8$$

$$\text{y: } E = 2^{n-1} - 1 = 2^7 - 1 = 127$$

Se requiere representar el número +31 en E_{127}

Por definición de la técnica de exceso:

$$+31_{E_{127}} = 31 + 127 = +158$$

Se codifica en binario el número +158:

$$+158 = 10011110_2$$

Es decir, que:

$$+31_{\text{E127}} = 10011110_2$$

Representación en Exceso $2^n - 1$

Ejemplo 2:

Supongamos que:

$$n=8$$

$$y: E = 2^{n-1} - 1 = 2^7 - 1 = 127$$

Se requiere representar el número -34 en E_{127}

Por definición de la técnica de exceso:

$$-34_{E_{127}} = -34 + 127 = +93$$

Se codifica en binario el número +93:

$$+93 = 01011101_2$$

Es decir, que:

$$-34_{E_{127}} = 01011101_2$$

Representación en Exceso $2^n - 1$

Cuando se quiere determinar el valor de un número binario representado en Exceso, se debe hacer la operación inversa, es decir restar el valor del Exceso E al número binario.

Ejemplo:

Supongamos que:

$$n=8$$

$$\text{y: } E = 2^{n-1} - 1 = 2^7 - 1 = 127$$

Se requiere determinar el valor de 10011110_{E127} (representado en E_{127})

$$10011110_2 = +158_{10}$$

Restando el Exceso $E = 127$:

$$+158 - 127 = +31$$

Por lo tanto:

$$10011110_{E127} = +31$$

Representación en Exceso

Resumen de las propiedades de la representación en Exceso:

➤ Rango de representación de E

$$-E \leq x \leq (2^n - 1) - E$$

➤ 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.

➤ Operaciones aritméticas más complejas, porque:

➤ Suma: $A_E + B_E = A + E + B + E$, es decir E está duplicado, y

➤ Resta: $A_E - B_E = A + E - (B + E)$, es decir E está eliminado

Sistemas Posicionales

Teorema fundamental de la numeración

Los sistemas numéricos se definen a partir de 2 conceptos:

- Símbolos, que forman parte del sistema
- Reglas, que definen como se generan los números

La cantidad de símbolos es finita, y define lo que se llama la Base del sistema numérico. Así, en el sistema decimal hay 10 símbolos, en el hexadecimal 16, en el binario 2, etc.

Las reglas establecen como se crean los números, en especial cuando se quieren representar más de los que forman la base del sistema. En general, la regla más importante es la que define el sistema como posicional. Un dígito tiene un “valor” no solo por el símbolo que es, sino por el lugar que ocupa en el número.

Sistemas Posicionales

El Teorema fundamental de la numeración establece que cualquier número en un sistema posicional, puede ser escrito como una suma finita de mintérminos compuestos por dígitos (pertenecientes al conjunto de símbolos que forman parte del sistema) multiplicados por la base elevada al exponente correspondiente a la posición que ocupa en el número.

De esta manera se simplifican los algoritmos requeridos para la ejecución de operaciones aritméticas tales como suma o resta.

Consideremos el siguiente ejemplo de un número decimal:

$$3574_{10}$$

En realidad, este número corresponde a lo siguiente:

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

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


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

Sistema posicional base decimal

Ejemplo en Base 10

Dígitos $\{0,1,2,3,4,5,6,7,8,9\}$

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

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


Sistema posicional base binario

Ejemplo en Base 2

Dígitos {0,1}

$$\begin{aligned} 1001,1_2 &= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} \\ &= 8 + 0 + 0 + 1 + 0,5 \\ &= 9,5_{10} \end{aligned}$$

Sistema posicional base 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

- Los sistemas numéricos de punto fijo son sistemas de representación de números reales, que consideran 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 el sistema decimal, los números podrían tener el siguiente formato de punto fijo:

0,23

5,12

9,11

En los 3 ejemplos anteriores, cada número tiene tres dígitos, y la coma está a la derecha del más significativo.

Números en punto fijo

- En el sistema binario, podemos representar números en punto fijo de una manera similar; es decir, podrían tener el siguiente formato

$$11,10 \longrightarrow (3,5)_{10}$$

$$01,10 \longrightarrow (1,5)_{10}$$

$$00,11 \longrightarrow (0,75)_{10}$$

- 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, en este último caso no se guarda la coma en ningún lugar, se supone que está en un lugar determinado.

Números en 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 = 0,00 a 9,99 es decir $[0,00...9,99]$
- Resolución = 0,01 porqué?

Porque la diferencia entre 2 puntos consecutivos de la representación es siempre la misma, y vale:

Ejemplo: $2,32 - 2,31 = 0,01$

Números en punto fijo

- Si mantenemos tres dígitos y desplazamos la coma hasta la derecha del último dígito, el rango y la resolución cambian de la siguiente manera:
 - Rango = 000 a 999 es decir [000...999]
 - Resolución = 1
- Aumenta el rango y disminuye la resolución. Es decir, dada una cantidad determinada de cifras en la representación, hay un compromiso entre rango y resolución.
- En cualquier caso, siempre hay 10^3 números distintos.

Números en punto fijo

Binario sin signo en punto fijo

4 dígitos parte entera

0 dígitos parte fraccionaria

- - - -

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

Números en punto fijo

Binario sin signo en punto fijo

3 dígitos parte entera
1 dígito parte fraccionaria

- - - , -

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

Números en punto fijo

Binario sin signo en punto fijo

2 dígitos parte entera

2 dígitos parte fraccionaria

- - , - -

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

Números en punto fijo

Binario sin signo en punto fijo

1 dígito parte entera

3 dígitos parte fraccionaria

- , - - -

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

Números en punto fijo

Binario sin signo en punto fijo

0 dígitos parte entera

4 dígitos parte fraccionaria

, - - - -

Rango: $[0 \dots 0,9375]$

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

Cuando hay que representar un número, podemos tener:

- Sin restricciones: se usa la cantidad de bits necesarios para representar el número. Por ejemplo, si se requiere representar el número 3.125, tengo la siguiente situación:

$$3,125_{10} = 11,001_2$$

Se usa la cantidad de bits de parte entera y fraccionaria necesarios para representar exactamente el número decimal $3,125_{10}$.

- Con restricciones: se define la cantidad de bits a asignar a parte entera y a parte fraccionaria. Por ejemplo, si se requiere representar 3,125 con 3 bits para parte entera y 4 bits para parte fraccionaria, resulta la siguiente situación:

$$3,125_{10} = 011,0010_2$$

Se deben completar con 0 los bits requeridos, tanto de la parte entera (0 a la izquierda) como de la parte fraccionaria (0 a la izquierda).

Representación y error

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

➤ 1) Sin error: existe el número binario que representa exactamente el número decimal.

Supongamos que queremos convertir el número decimal $3,125_{10}$. Se puede representar de la siguiente manera:

$$3,125_{10} = 11,001_2$$

Representación y error

➤ 2) Con error: se debe usar el número binario más próximo posible (menor error) al número decimal. Supongamos que queremos convertir el número decimal $3,2_{10}$.

➤ Si se usan 3 bits para parte fraccionaria, se tiene:

▶ $011,001_2 = 3,125_{10}$

$$\text{Error} = 3,2_{10} - 3,125_{10} = 0,075_{10}$$

▶ $011,010_2 = 3,250_{10}$

$$\text{Error} = 3,2_{10} - 3,25_{10} = -0,05_{10}$$

➤ Si se usan 4 bits para parte fraccionaria, se tiene:

▶ $011,0011_2 = 3,1875_{10}$

$$\text{Error} = 3,2_{10} - 3,1875_{10} = 0,0125_{10}$$

➤ Si se usan 5 bits para parte fraccionaria, se tiene:

▶ $011,00111_2 = 3,21875_{10}$

$$\text{Error} = 3,2_{10} - 3,21875_{10} = -0,01875_{10}$$

Por más que se agreguen bits decimales no se puede conseguir eliminar el error de conversión.

Representación y error

- En el ejemplo anterior, el número binario que debería usarse es $011,0011_2$ que es igual a $3,1875_{10}$.
- Es decir, en lugar de almacenar $3,2_{10}$ se deberá usar $3,1875_{10}$, cometiéndose un error igual a $0,0125_{10}$.

Suma y resta en binario

- Una suma en binario se realiza en forma similar a una suma decimal.
- Se suman dígito a dígito (es decir bit a bit) los n dígitos (los n bits) de cada número, empezando por el 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.
- Vamos a ver a continuación 2 ejemplos de sumas en decimal.

Suma y resta en binario

75

➤ Ejemplo 1: suma de 2 números decimales de 1 cifra, y el resultado entra en 1 cifra:

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

➤ Ejemplo 2: suma de 2 números decimales de 1 cifra, y el resultado necesita 2 cifras (no "entra" en 1 cifra):

$$\begin{array}{r} 1 \\ 5 \\ + 6 \\ \hline 11 \end{array}$$

Suma y resta en binario

- Ejemplo 1: suma de 2 números binarios de 4 cifras:

$$\begin{array}{r} 0100 \\ + 0010 \\ \hline 0110 \end{array}$$

- Ejemplo 2: suma de 2 números binarios de 4 cifras:

$$\begin{array}{r} 1111 \\ + 1101 \\ \hline 0000 \end{array}$$

Como tener en cuenta esta situación?

Suma y resta en binario

- Cuando se suman o restan 2 números es útil detectar ciertas situaciones resultantes de la operación.
- Por ejemplo:
 - El resultado es cero?
 - El resultado es negativo?
 - El resultado es correcto?
- Una situación que puede ocurrir es que el resultado no “entre” en la cantidad de bits usados para la representación.
- Que un resultado no “entre” en la representación significa que el resultado requiere más bits de los usados para poder representarlo. Esta situación es idéntica a la que ocurre cuando operamos en decimal.

Bits (banderas) de condición

- Para atender las situaciones que se presentan en las operaciones aritméticas en un sistema de cómputo se requiere disponer de unos bits extra (además de los operandos y el resultado) denominados BITS DE CONDICIÓN (también llamados “flags” o banderas de estado) que reflejen el resultado real de una operación aritmética.
- Sus valores (“estados”) permiten tomar decisiones referidas al resultado de una operación aritmética. Por ejemplo, decidir qué acción ejecutar si el resultado de la operación aritmética resultó 0, o distinta de 0.
- Típicamente, los procesadores disponen de varias banderas de estado.

Bits (banderas) de condición

➤ Bandera de cero (Z)

- $Z = 1$ si el resultado de la operación es igual a 0.
- $Z = 0$ si el resultado de la operación es igual a $\neq 0$

La bandera de cero se “activa” (es decir se pone en 1) si el resultado de la operación aritmética involucrada dio 0.

Bits (banderas) de condición

➤ Bandera de arrastre o “carry” (C)

- En sumas: $C = 1$ si hay acarreo del bit más significativo
- En restas $C = 1$ si hay pedido (‘borrow’) hacia el bit más significativo.

Observación: cuando la operación involucra números sin signo, $C=1$ indica una condición de desborde o fuera de rango (es decir, el resultado no entra en la cantidad de bits de la representación).

Bits (banderas) de condición

➤ Bandera de negativo (N):

- $N = 1$ si el resultado de la operación es negativo
- $N = 0$ si el resultado de la operación es positivo

Observación 1: la bandera supone que el número está en una representación con signo (CA2) donde el bit más significativo identifica el signo del número.

En otras palabras:

$N = 1$ si el bit más significativo del resultado es 1

Observación 2: si los números con los que se está trabajando son sin signo, no tiene sentido la bandera N.

Bits (banderas) de condición

➤ Bandera de desborde u overflow (V)

- $V = 1$ si el resultado de la operación está “fuera de rango”
- $V = 0$ si el resultado de la operación no está “fuera de rango”

Observación: la bandera V indica que el resultado de una operación aritmética en CA2 no se puede expresar con el número de bits utilizado (“no entra” o está fuera del rango representable en la cantidad de bits usados).

Hay 2 situaciones para identificar:

- 1) Suma
- 2) Resta

Bits (banderas) de condición

1) Bandera de desborde en la suma

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.

Es decir, la lógica que genera la bandera V hace lo siguiente:

Bits (banderas) de condición

Lógica de la bandera V en operaciones de suma

$V = 1$ (es decir hay overflow en la suma) si:

- Suma de 2 números + y el resultado es –
o

- Suma de 2 números – y el resultado es +

y:

$V = 0$ (es decir no hay overflow en la suma) si:

- Suma de 2 números de distinto signo.

Bits (banderas) de condición

2) Bandera de desborde en la resta

Ocurre cuando al restar 2 números, el resultado no entra en la cantidad de bits usados.

➤ Cuando hay Overflow en la resta?

- Restando a un número + un número - y el resultado es -
- Restando a un número - un número + y el resultado es +

➤ Cuando NO hay Overflow en la resta?

- Restando 2 números de igual signo.

Bits (banderas) de condición

Lógica de la bandera V en operaciones de resta

$V = 1$ (es decir hay overflow en la resta) si:

- Resta a un número + un – y el resultado es – o

- Resta a un número – un + y el resultado es +

y:

$V = 0$ (es decir no hay overflow en la resta) si:

- Resta de números de igual signo.

Bits (banderas) de condición

87

A continuación se verán varios ejemplos de operaciones aritméticas, y el análisis a hacer respecto de los valores que toman las banderas de estado en cada caso.

Bits de condición para la suma

Operación	NZVC	Ca2	BSS
$\begin{array}{r} 0100 \\ + 0010 \\ \hline 0110 \end{array}$	 0 0 0 0	$\begin{array}{r} +4 \\ + 2 \\ \hline +6 \end{array}$	$\begin{array}{r} +4 \\ + 2 \\ \hline +6 \end{array}$

✓ Ca2: correcto

✓ BSS: correcto

Bits de condición para la suma

Operación NZVC Ca2 BSS

$$\begin{array}{r} + \quad 0101 \\ \quad 0111 \\ \hline 1100 \end{array}$$

$$\begin{array}{r} 1010 \\ + \quad +5 \\ \quad +7 \\ \hline -4 \end{array}$$

$$\begin{array}{r} + \quad +5 \\ \quad +7 \\ \hline +12 \end{array}$$

✓ Ca2: incorrecto

✓ BSS: correcto



Bits de condición para la suma

Operación	NZVC	Ca2	BSS
-----------	------	-----	-----

$ \begin{array}{r} + 1101 \\ 0011 \\ \hline 1 \ 0000 \end{array} $	$ \begin{array}{r} 0 \ 1 \ 0 \ 1 \end{array} $	$ \begin{array}{r} + -3 \\ +3 \\ \hline 0 \end{array} $	$ \begin{array}{r} + 13 \\ 3 \\ \hline 0 \end{array} $
---	--	--	---

✓ Ca2: correcto

✓ BSS: incorrecto

Bits de condición para la suma

Operación	NZVC	Ca2	BSS
$ \begin{array}{r} +1001 \\ 1100 \\ \hline 0101 \end{array} $	0 0 1 1	$ \begin{array}{r} -7 \\ + -4 \\ \hline +5 \end{array} $	$ \begin{array}{r} 9 \\ + 12 \\ \hline 5 \end{array} $

✓ Ca2: incorrecto

✓ BSS: incorrecto

Bits de condición para la resta

Operación	NZVC	Ca2	BSS
-----------	------	-----	-----

$$\begin{array}{r}
 1001 \\
 \underline{0100} \\
 0101
 \end{array}$$

0 0 1 0

$$\begin{array}{r}
 -7 \\
 \underline{+4} \\
 +5
 \end{array}$$

$$\begin{array}{r}
 9 \\
 \underline{4} \\
 5
 \end{array}$$

✓ Ca2: incorrecto

✓ BSS: correcto



Bits de condición para la resta

Operación	NZVC	Ca2	BSS
$ \begin{array}{r} 1 \quad 0101 \\ \underline{\quad} \\ \quad 0111 \\ \hline \quad 1110 \end{array} $	1 0 0 1	$ \begin{array}{r} +5 \\ \underline{+7} \\ -2 \end{array} $	$ \begin{array}{r} 5 \\ \underline{7} \\ 14 \end{array} $

- ✓ Ca2: correcto
- ✓ BSS: incorrecto

Suma en Binario con signo (BCS)

Como sería una operación aritmética de números binarios con signo (BCS)?

Ejemplo: se tienen que sumar los siguientes números en BCS:

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

Como sería el algoritmo de suma?

Otras representaciones

Existen otras representaciones que se usan en sistemas de cómputo. Por ejemplo:

- BCH
- BCD
- Caracteres

Sistema hexadecimal codificado en binario (BCH)

96

Representación en BCH:

- En el sistema hexadecimal se tienen 16 dígitos (0-9, A, B, C, D, F).
- Cada dígito hexadecimal requiere 4 bits para representarlo en binario.
- Para pasar de hexadecimal a binario se convierte cada dígito hexadecimal en 4 bits.
- Por ejemplo:

$$\text{ADF4}_{16} = 1010\ 1101\ 1111\ 0100_2$$

BCH

97

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)

Representación en BCD:

- En el sistema decimal se tienen 10 dígitos (0-9).
- Por lo tanto, cada dígito decimal requiere 4 bits para representarlo en binario.
- El problema surge porque con 4 bits tengo 16 combinaciones (representaciones) distintas, de las cuales solo se están usando 10. El resto de las combinaciones no son números decimales (BCD).

BCD

99

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

10

0

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

10

1

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 desempquetado.

➤ 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

10

4

Ejemplo:

$$+ 834 = 11111000 \ 11110011 \ 11000100 = F8 \ F3 \ C4$$

$$- 834 = 11111000 \ 11110011 \ 11010100 = F8 \ F3 \ D4$$

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

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:

$$+ \text{834} = 10000011 \text{ 01001100} = \text{83 4C}$$

$$- \text{34} = 00000011 \text{ 01001101} = \text{03 4D}$$

Suma en BCD

10

6

- 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

10
7

➤ Suma ≤ 9

decimal

BCD

+ 41
22

+ 0100 0001
0010 0010

63

0110 0011

El resultado es correcto porque ambas cifras del resultado son menores que 9.

Suma en BCD

➤ Suma > 9

decimal

BCD

1

111

+ 15

+ 0001 0101

+

27

+

0010 0111

42

0011 1100

3

no válido

Suma en BCD

109

- Dado que en BCD no existen las cifras mayores a 9, cuando la suma de algún dígito da >9 significa que el resultado está mal. La máquina operó en BCH sin tener en cuenta que los números eran BCD.
- Sin embargo, se puede compensar (ajustar) el resultado para llevarlo a BCD, teniendo en cuenta la “distancia” entre las representaciones BCH y BCD. Esa distancia entre ambas representaciones es igual a 6.
- La corrección a BCD se consigue sumando 6 a las cifras que hayan resultado ser mayores que 9.
- Conclusión: cuando la suma de los dígitos es > 9 hay que sumar 6 en ese dígito.

Suma en BCD

110

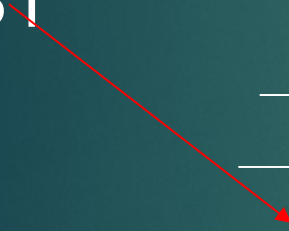
Ejemplo 1:

$ \begin{array}{r} + \overset{1}{15} \\ 27 \\ \hline 42 \end{array} $	$ \begin{array}{r} + \overset{111}{0001\ 0101} \\ 0010\ 0111 \\ \hline 0011\ 1100 \\ + 0110 \\ \hline 0100\ 0010 \end{array} $	<p>sumar 6 a esta cifra</p> <p>← Resultado corregido</p>
--	--	--

Suma en BCD

Ejemplo 2:

476	0100 0111 0110	
+ 55	+ 0101 0101	
<hr/>	<hr/>	
531	0100 1100 1011	
	+ 0110 0110	
	<hr/>	
	0101 0011 0001	← Resultado corregido
	5 3 1	



Representación alfanumérica

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

Representación alfanumérica

A cada símbolo un código en binario:

Ejemplo: x, y, α , β , #, @, [,]

Si tengo 8 símbolos ¿Cuántos bits? ¿Por qué?

000 x

001 y

010 α

011 β

100 #

101 @

110 [

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

- 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

116

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		DEL

Una extensión al ASCII

117

128	Ç	144	É	160	á	176	☐	193	⊥	209	⌞	225	β	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	â	150	û	166	°	182	⌞	199	⌞	215	⌞	231	τ	247	≈
135	ç	151	ù	167	°	183	⌞	200	⌞	216	⌞	232	Φ	248	°
136	ê	152	—	168	¿	184	⌞	201	⌞	217	⌞	233	⊗	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	¾	188	⌞	205	=	221	■	237	φ	253	z
141	ï	158	—	173	j	189	⌞	206	⌞	222	■	238	ε	254	■
142	Ä	159	f	174	«	190	⌞	207	⌞	223	■	239	∧	255	
143	Å	192	L	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