

Organización de Computadoras



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***, 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/>



Evaluación de cursado.

- No se registra asistencia a Prácticas pero se debe estar Inscripto en sistema SIU-Guarani.
- Evaluación en TRES etapas PARCIALES
 - Parcial 1 : Evaluación de Prácticas 1 y 2.
 - Parcial 2 : Evaluación de Prácticas 3 y 4.
 - Parcial 3 : Evaluación de Práctica 5.

Cada Parcial tiene UN (1) Recuperatorio

Alumno Desaprobado ó Ausente dos veces en el mismo Parcial podrá rendir Evaluación Total al final del curso



Parciales de Prácticas. Fechas.

Anotación previa para rendir parciales: por WebUNLP ó presencial el Jueves 16 de ABRIL

- Fechas (la ausencia se considera Desaprobado)

- Jueves 23 de ABRIL: Parcial 1
 - Jueves 07 de Mayo: Rec Parcial 1
- Jueves 21 de MAYO: Parcial 2
 - Jueves 11 de Junio: Rec Parcial 2
- Jueves 25 de JUNIO: Parcial 3
 - Viernes 10 de Julio: Rec Parcial 3

Anotación previa para rendir TOTAL: por WebUNLP ó presencial el Martes 14 de JULIO, 18:30 hs, Aula1

- Jueves 16 de JULIO: Evaluación TOTAL



Promoción de asignatura. Condiciones y Fechas.

Aprobar Parciales 1 y 2 en primera fecha con el 70% de la nota máxima habilita para rendir

- Evaluación Corta de Teoría (ETC)

- VIERNES 29 de MAYO

Aprobar la ETC y el Parcial 3 en primera fecha con el 70% de la nota máxima habilita para rendir

- Evaluación Teórica Promoción

- VIERNES 03 de JULIO



Conceptos básicos

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



Representación de datos

- Las computadoras almacenan datos e instrucciones en memoria
- Para ello utilizan el sistema binario
- Razones :
 - el dispositivo se encuentra en uno de dos estados posibles (0 ó 1)
 - identificar el estado es más fácil si sólo hay dos



Representación de datos

- 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 4 tipos básicos de datos binarios

- Números enteros sin/con signo
- Números reales con signo
- Números decimales codificados en binario (BCD)
- Caracteres



Representación de números enteros

- Sin signo (BSS)
- Con signo
 - 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

Si el número tiene n bits, puedo representar

➤ $2^n =$ números distintos

El rango va desde

➤ 0 a $(2^n - 1)$



Números enteros sin signo

Ejemplo: $n = 3$ bits

Decimal	Representación sin signo
0	000
1	001
2	010
..
7	111



Números enteros sin signo

Ejemplo: $n = 8$ bits

0	00000000
..
128	10000000
..
254	11111110
255	11111111



Números enteros sin signo

- RECORDAR: la cantidad de representaciones distintas depende del número de bits

$$\text{N}^{\circ}\text{s distintos} = 2^n$$





Sistemas Posicionales

Teorema Fundamental de la Numeración

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

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

N° es el valor decimal de una cantidad expresada en base B y con $(n+1+m)$ dígitos en posiciones i .



Sistema Decimal

- Base 10.
- Dígitos $\{0,1,2,3,4,5,6,7,8,9\}$

$$3574 = 3000 + 500 + 70 + 4$$

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

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

$$3.1416_{(10)} = 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2} + 1 \times 10^{-3} + 6 \times 10^{-4}$$

- 3 unidades + 1 décima + 4 centésimas + 1 milésima + 4 diezmilésimas



Sistema Binario

- 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 Hexadecimal

- 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

$$\begin{aligned} 2CA,8_{16} &= 2 \times 16^2 + C \times 16^1 + A \times 16^0 + 8 \times 16^{-1} \\ &= 512 + 192 + 10 + 0,5 \\ &= 714,5_{10} \end{aligned}$$



Números enteros con signo

- BCS
- Técnica de Complementos
- Técnica de Exceso



Representación en BCS

- Con n bits, 1 bit representa al signo y $n-1$ bits a la magnitud



- El bit $n-1$ (extremo izquierdo) representa sólo al signo
- Los bits 0 a $n-2$ la magnitud



Binario con signo

- Un 0 en el bit de signo indica que el número es positivo
- Un 1 en el bit de signo indica que el número es negativo
- Los bits $0 \rightarrow n-2$ representan el valor absoluto en binario
- El rango: $-(2^{n-1} - 1) \rightarrow +(2^{n-1} - 1)$ con 2 ceros

Binario con signo (2)

➤ Ejemplos

$$+32_{10} = \overset{+}{00100000}$$

↓
32

$$-32_{10} = \overset{-}{10100000}$$

↓
32

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

$$-7_{10} = 10000111$$

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

$$-41_{10} = 10101001$$



Binario con signo (3)

➤ Ejemplo: $n=8$ bits

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



Binario con signo (4)

➤ Ejemplo con $n = 3$ bits

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

$$110 = -2$$

$$101 = -1$$

$$100 = -0$$

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

$$010 = +2$$

$$001 = +1$$

$$000 = +0$$



Resumen: BCS

- ✓ El intervalo es simétrico
- ✓ El primer bit sólo indica el signo
- ✓ Los positivos empiezan con cero (0)
- ✓ Los negativos empiezan con uno (1)
- ✓ Hay dos ceros
- ✓ Números distintos: 2^n



Técnica de Complementos

- El complemento a un número N de un número A (A menor que N) es igual a la cantidad que le falta a A para ser N

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

- El complemento a un número N del número $(N-A)$ es igual a A .

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



Técnica de Complementos (2)

En un sistema con n dígitos podemos tener:

- Complemento a la base disminuida

- si $N = \text{base}^n - 1$

En sistema binario es **Complemento a 1** ó **Ca1**

- Complemento a la base

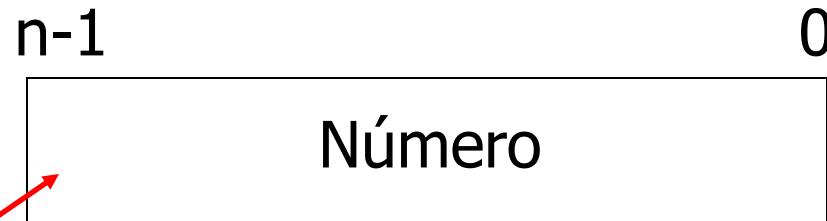
- si $N = \text{base}^n$

En sistema binario es **Complemento a 2** ó **Ca2**



Representación en Ca1

❖ Los n bits representan al número



❖ Información del signo



Ca1

- Si el número es positivo, los n bits tienen la representación binaria del número (como siempre)
- Si el número es negativo, los n bits tienen el Ca1 del valor deseado.
- El Ca1 de un número en base 2 se obtiene invirtiendo todos los bits



Ca1

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- El rango va desde
 $-(2^{n-1} - 1)$ a $+(2^{n-1} - 1)$
con dos ceros



Ca1

➤ Ejemplos

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

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

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

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

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

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



Ca1

➤ Ejemplo: $n=8$ bits

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



Ca1

➤ Ejemplo con $n = 3$ bits

$$111 = -0$$

$$110 = -1$$

$$101 = -2$$

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

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

$$010 = +2$$

$$001 = +1$$

$$000 = +0$$



Ca1

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

❖ Cuando es positivo:

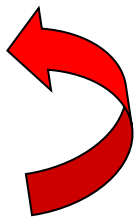
$$01100000 = 1 \times 2^6 + 1 \times 2^5 = 64 + 32 = 96$$

Como siempre



Ca1

- ❖ Cuando es negativo, puedo hacer dos cosas:
- ✓ Ca1 del número y obtengo el positivo
Ej.

$$\begin{array}{lcl} 11100000 & = - & 31 \\ 11100000 & \xrightarrow{\text{red arrow}} & 00011111 = +31 \end{array}$$




Ca1

- ✓ Otro método: el peso que tiene el primer dígito ahora es $-(2^{n-1} - 1)$ y el resto de los dígitos con pesos positivos *como siempre*

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

- *O por definición de Complemento a la base disminuida*

$$\text{Ca1} = (b^n - 1) - N^o$$





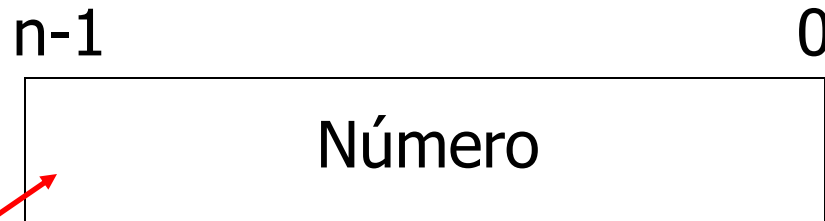
Resumen Ca1

- ❖ El intervalo es simétrico
- ❖ Los n bits representan al número
- ❖ Los positivos empiezan con cero (0)
- ❖ Los negativos empiezan con uno (1)
- ❖ Hay dos ceros
- ❖ Números distintos 2^n



Representación en Ca2

❖ Los n bits representan al número



❖ Información del signo



Representación en Ca2

- Si el número es positivo, los n bits tienen la representación binaria del número (como siempre)
- Si el número es negativo, los n bits tienen el Ca2 del valor deseado.
- El Ca2 de un número (en base 2) se obtiene invirtiendo todos los bits (Ca1) y luego sumándole 1.



Ca2

- Otra forma: “mirando” desde la derecha se escribe el número (base 2) igual hasta el primer “1” uno inclusive y luego se invierten los demás dígitos
- Otra forma: *por definición de Complemento a la base*

$$\text{Ca2} = b^n - N^0 \quad \leftarrow$$



Ca2

- Los positivos empiezan con cero (0)
- Los negativos empiezan con uno (1)
- El rango es asimétrico y va desde
 $-(2^{n-1})$ a $+(2^{n-1} - 1)$
- Hay un solo cero



Ca2

➤ Ejemplos

$$+32_{10} = 00100000 \leftarrow \text{"mirando" desde la derecha}$$

$$-32_{10} = 11100000$$

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



Ca2 (otra forma)

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

1111

11011111 invierto todos los bits

+ 1 le sumo 1

$$-32_{10} = 11100000 \quad \leftarrow \text{ en Ca2}$$



Ca2 (otra forma)

- $\text{Ca2} = b^n - N^o = 2^8 - 32 = 256 - 32 = 224$
- Hagamos la cuenta en base 2

$$\begin{array}{r} 011 \\ \text{---} 1101000000 \\ \text{---} 001000000 \\ \hline -32 = 111000000 \end{array} \quad \leftarrow \text{ en Ca2}$$



Ca2

➤ Ejemplo : $n=8$ bits

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



Ca2

➤ Ejemplo con $n = 3$ bits

$$111 = -1$$

$$110 = -2$$

$$101 = -3$$

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

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

$$010 = +2$$

$$001 = +1$$

$$000 = +0$$



Ca2

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

❖ Cuando es positivo:

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

Como siempre



Ca2

❖ Cuando es negativo, puedo hacer dos cosas:

✓ Ca2 el número y obtengo el positivo

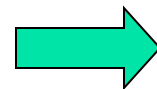
Ej.

11100000

= - 32



11100000



00100000 = +32



Ca2

- ✓ Otro método: el peso que tiene el primer dígito ahora es $-(2^{n-1})$ y el resto de los dígitos con pesos positivos *como siempre*

$$\begin{aligned} 11100000 &= -1 \times (2^7) + 1 \times 2^6 + 1 \times 2^5 \\ &= -128 + 64 + 32 = -32 \end{aligned}$$



Resumen Ca2

- ❖ El intervalo es asimétrico, hay un - más
- ❖ Los n bits representan al número
- ❖ Los positivos empiezan con cero (0)
- ❖ Los negativos empiezan con uno (1)
- ❖ Hay un solo cero
- ❖ Números distintos 2^n



Técnica de Exceso

- La representación de un número A es la que corresponde a la SUMA del mismo y un valor constante E (o exceso).

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

- Dado un valor, el número representado se obtiene RESTANDO el valor del exceso.

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

- El signo del número A resulta de una resta
 - En binario, NO sigue la regla del bit mas significativo



Exceso 2^{n-1}

■ Rango

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

si $n=6$ **Exceso 32**

$$\begin{aligned} -2^{(6-1)} &= 000000_2 \\ &= \mathbf{0 - 32} \end{aligned}$$

$$\begin{aligned} 2^{(6-1)}-1 &= 111111_2 \\ &= \mathbf{63 - 32 = 31} \end{aligned}$$

$$\begin{aligned} 0_{10} &= 100000_2 \\ &= \mathbf{32 - 32 = 0} \end{aligned}$$



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: 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 2^{do} y 3^{er} 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

- Para el ejemplo anterior en sistema decimal

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

Resolución es 0,01

$$2,32 - 2,31 = 0,01 \quad \text{o} \quad 9,99 - 9,98 = 0,01$$



Rango y Resolución(2)

- ❖ Notar que hay un compromiso entre rango y resolución.
- ❖ Si mantenemos tres dígitos y desplazamos la coma dos lugares a la derecha, el rango pasa a ser $[0, \dots, 999]$ y la resolución valdrá 1.

En cualquiera de los casos hay 10^3 números distintos



Ejemplo en BSS con 4 bits

4 parte ent. y 0 parte frac.

- - - -

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.

- - - , -

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.

- - , - -

Resolución

$$00,01 - 00,00 =$$

$$00,01_2 = 0,25_{10}$$

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.

- , - - -

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.

, - - - -

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 sistema binario tendremos 2 casos:
 - 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
 - $3,125_{10} = 011,0010_2$

No cometemos error



Representación y error (2)

- Convertir $3,2_{10}$ con distintas restricciones
 - 3 bits para parte fraccionaria: $011,001_2 = 3,125_{10}$
 - Error = $3,2 - 3,125 = 0,075$
 - 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 = $3,2 - 3,21875 = -0,01875$
- El error más pequeño es 0,0125 entonces 3,1875 es la representación más cercana a 3,2 y podría utilizar sólo 4 bits para la parte fraccionaria.



Bits de condición (banderas)

- ✓ Son bits que el procesador establece de modo automático acorde al resultado de cada operación realizada.
- ✓ Sus valores permitirán tomar decisiones como:
 - ✓ Realizar o no una transferencia de control.
 - ✓ Determinar 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

- Para sumar dos números en Ca2 se suman los n bits directamente.
- Si sumamos dos números $+$ y el resultado es $-$ ó si sumamos dos $-$ y el resultado es $+$ **hay overflow**, en otro caso no lo hay.
- Si los N^os son de distinto signo nunca puede haber overflow.



Resta en Ca2

- Para restar dos números en Ca2, se restan los n bits directamente. También se puede Ca2 el sustraendo y transformar la resta en suma.
- Si a un $N^0 +$ le restamos un $N^0 -$ y el resultado es $-$ ó si a un $N^0 -$ le restamos un $+$ y el resultado es $+$ hay overflow en la resta.
- Si son del mismo signo nunca hay overflow



Bits de condición para la suma

Operación	NZVC	Ca2	Sin signo
-----------	------	-----	-----------

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

0000

$$\begin{array}{r} + \quad +4 \\ \quad +2 \\ \hline +6 \end{array}$$

$$\begin{array}{r} + \quad +4 \\ \quad +2 \\ \hline +6 \end{array}$$

✓ Los dos resultados son correctos.



Bits de condición para la suma

Operación	NZVC	Ca2	Sin signo
-----------	------	-----	-----------

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

1010

$\begin{array}{r} + 5 \\ + 7 \\ \hline \end{array}$	$\begin{array}{r} + 5 \\ + 7 \\ \hline \end{array}$
-4 overf.	+12

✓ Ca2 incorrecto, sin signo correcto.



Bits de condición para la suma

Operación	NZVC	Ca2	Sin signo
-----------	------	-----	-----------

$$\begin{array}{r} + 1101 \\ + 0011 \\ \hline 1 \leftarrow 0000 \end{array}$$

0101

$$\begin{array}{r} + -3 \\ + +3 \\ \hline 0 \end{array}$$

$$\begin{array}{r} + 13 \\ + 3 \\ \hline \text{carry } 0 \end{array}$$

✓ Ca2 correcto, sin signo incorrecto.



Bits de condición para la suma

Operación	NZVC	Ca2	Sin signo
$\begin{array}{r} + \\ 1001 \\ 1100 \\ \hline 1 \leftarrow 0101 \end{array}$	0011	$\begin{array}{r} + \\ -7 \\ -4 \\ \hline V +5 \end{array}$	$\begin{array}{r} + \\ 9 \\ 12 \\ \hline C 5 \end{array}$

✓ Los dos resultados son incorrectos.



Bits de condición para la resta

Operación	NZVC	Ca2	Sin signo
$\begin{array}{r} 1 \rightarrow 0101 \\ \underline{0111} \\ 1110 \end{array}$	1001	$\begin{array}{r} +5 \\ \underline{+7} \\ -2 \end{array}$	$\begin{array}{r} 5 \\ \underline{7} \\ \text{B } 14 \end{array}$

✓ Ca2 correcto, sin signo incorrecto.



Bits de condición para la resta

Operación	NZVC	Ca2	Sin signo
$\begin{array}{r} 1001 \\ - 0100 \\ \hline 0101 \end{array}$	0010	$\begin{array}{r} -7 \\ +4 \\ \hline \text{V} +5 \end{array}$	$\begin{array}{r} 9 \\ - 4 \\ \hline 5 \end{array}$

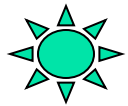
✓ Ca2 incorrecto, sin signo correcto.



Suma en BCS

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

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



Pensar el algoritmo de suma.



Otras representaciones

- BCH
- BCD
- Caracteres



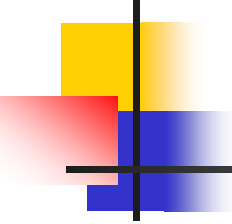
Sistema hexadecimal codificado en binario (BCH)

- 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

- BCD tiene dos ámbitos de aplicación:
 - E/S y periféricos, los números se codifican usando un byte por dígito. Se dice que el número está *desempaquetado*.
 - En cálculo, se reservan 4 bits por dígito. Se dice que el número está *empaquetado*.



BCD

- Ejemplo: desempaquetado sin signo

$$\begin{aligned} 834 &= 11111000 \ 11110011 \ 11110100 \\ &= F8 \ F3 \ F4 \end{aligned}$$

- Por cada dígito se usan 8 bits, 4 para el binario puro y 4 se completan con "1"



BCD

- Desempaquetado con signo
- Con 4 bits hay $2^4=16$ combinaciones posibles de unos y ceros :
 - Diez usamos para los dígitos 0 al 9
 - Nos quedan seis sin usar
 - $C_{16} = 1100$ representa al signo +
 - $D_{16} = 1101$ representa al signo -



BCD

- Ejemplo: desempaquetado con signo

$$\begin{aligned} + 834 &= 11111000 \ 11110011 \ 11000100 \\ &= F8 \ F3 \ C4 \end{aligned}$$

- Los 4 bits que acompañan al último dígito son reemplazados por el signo.



BCD

Ejemplo:

$$\begin{aligned} \blacksquare - 834 &= 1111\textcolor{red}{1}000 \ 1111\textcolor{green}{0}011 \ 1101\textcolor{blue}{0}100 \\ &= \textcolor{red}{F}8 \ \textcolor{green}{F}3 \ \textcolor{blue}{D}\textcolor{yellow}{4} \end{aligned}$$



BCD

Ejemplo: empaquetado con signo

$$\blacksquare + 834 = 10000011 \ 01001100$$

$$= 83 \ 4C$$

$$\blacksquare - 34 = 00000011 \ 01001101$$

$$= 03 \ 4D$$



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

- En el primer caso no hay problema

$$\begin{array}{r} + \quad 41 \\ \quad 22 \\ \hline 63 \end{array}$$

$$\begin{array}{r} + \quad 0100 \quad 0001 \\ \quad 0010 \quad 0010 \\ \hline 0110 \quad 0011 \end{array}$$



Suma en BCD

- En el segundo caso ¿Qué sucede ?

$$\begin{array}{r} 1 \\ + 15 \\ + 27 \\ \hline 42 \end{array}$$

$$\begin{array}{r} 111 \\ + 0001 \ 0101 \\ + 0010 \ 0111 \\ \hline 0011 \ 1100 \\ \underbrace{\hspace{1cm}}_3 \quad \underbrace{\hspace{1cm}}_{\text{no válido}} \end{array}$$



Suma en BCD

- Cuando la suma de los dos dígitos da >9 hay que generar el “acarreo” porque hay seis combinaciones no usadas
- Entonces: cuando la suma de los dígitos es > 9 hay que sumar 6 en ese dígito



Suma en BCD

$$\begin{array}{r} 1 \\ + 15 \\ + 27 \\ \hline 42 \end{array}$$

$$\begin{array}{r} 111 \\ + 0001 \ 0101 \\ + 0010 \ 0111 \\ \hline 111 \ 1 \\ 0011 \ 1100 \\ + \ 0110 \quad \leftarrow 6 \\ \hline 0100 \ 0010 \end{array}$$

Resultado correcto



Suma en BCD

- Ejemplo

$$\begin{array}{r} 1 \\ 476 \\ + 55 \\ \hline 531 \end{array}$$

$$\begin{array}{r} 111 \quad 1 \\ 0100 \quad 0111 \quad 0110 \\ + \quad 0101 \quad 0101 \\ \hline 0100 \quad 1100 \quad 1011 \\ + \quad 0110 \quad 0110 \\ \hline 0101 \quad 0011 \quad 0001 \end{array}$$



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		DEL

Una extensión al ASCII

128	Ç	144	É	160	á	176	░	193	⌞	209	⌞	225	β	241	±
129	ù	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	ª	182	⌤	199	⌤	215	⌤	231	τ	247	≈
135	ç	151	ù	167	º	183	⌥	200	⌥	216	⌥	232	⊕	248	°
136	ê	152	—	168	¿	184	⌦	201	⌦	217	⌦	233	⊗	249	·
137	ë	153	Ö	169	—	185	⌧	202	⌧	218	⌧	234	Ω	250	·
138	è	154	Û	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	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