Universidad Tecnológica Nacional
Facultad Regional Tucumán
Ingeniería en Sistemas de Información
ARQUITECTURA DE COMPUTADORES

TRABAJO PRÀCTICO Nº 2

SISTEMAS DE NUMERACIÓN POSICIONAL.
SUMAS.
RESTAS USANDO COMPLEMENTO A LA BASE Y A LA BASE MENOS UNO.
MULTIPLICACIONES.
OTROS CÓDIGOS.

SISTEMAS DE NUMERACIÓN POSICIONAL

Características de los SNP

- Usan símbolos distintos.
- Tienen el valor nulo o cero.
- La base indica la cantidad de símbolos distintos que poseen.
- El primer número que se escriba con dos dígitos es la base.
- La posición de cada dígito en el número es importante.

¿Cuál es el sistema de numeración posicional que usan habitualmente?

Base 10 o Decimal

Símbolos distintos

BASE 10
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Valor nulo o cero

Base, el 1º que se escribe con dos dígitos

...

$$35 \neq 53$$

...

La posición de cada dígito es importante

Sistema Binario o Base 2

Un bit es la unidad mínima de información, un 0 o un 1.

Los datos viajan en las computadoras por cables, a través de impulsos eléctricos, que son representados por dos estados:

- Prendido, Abierto, 1.
- Apagado, Cerrado, 0.

Las computadoras utilizan el sistema binario, 0 y 1, para representar toda la información.

Sistema Binario o Base 2

BASE 2
0
1
10

Base, el 1º que se escribe con dos dígitos

Sistema Binario o Base 2

BASE 2
0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1110
1111

…

…

Sistemas que se relacionan con el Binario o Base 2

BASE 16 → $2^4 = 16$

BASE 8 → $2^3 = 8$

Sistema Hexadecimal o Base 16

BASE 16 → $2^4 = 16$
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

10 → Base, el 1º que se escribe con dos dígitos

…

…

Sistema Octal o Base 8

BASE 8 → $2^3 = 8$
0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17

...

Equivalencia entre los sistemas

BASE 2	BASE 8 → $2^3 = 8$	BASE 10	BASE 16 → $2^4 = 16$
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Poner como subíndices la base a la que pertenece el número

$$10_{(10)} \neq 10_{(2)} \neq 10_{(8)} \neq 10_{(16)}$$

EJEMPLO: Sistema de Base 4

Símbolos del Sistema de Base 4: + - / *

BASE 4
+
-
/
*
-+
--
-/
-*
/+
/-
//
/*

CAMBIOS DE BASE ENTRE SISTEMAS

Cambios de base

Posición = X	7	6	5	4	3	2	1	0	-1	-2	-3	-4
Valor = 2^X	128	64	32	16	8	4	2	1	**0,5**	**0,25**	**0,125**	**0,0625**

ESTOS VALORES LOS DEBEN RECORDAR!

Cambios de base

Ejercicio:

Dado el número binario, pasarlo a **base 10**:

64 32 16 8 4 2 1 $\Rightarrow$ Valores de las posiciones

$$0101111_{(2)}$$

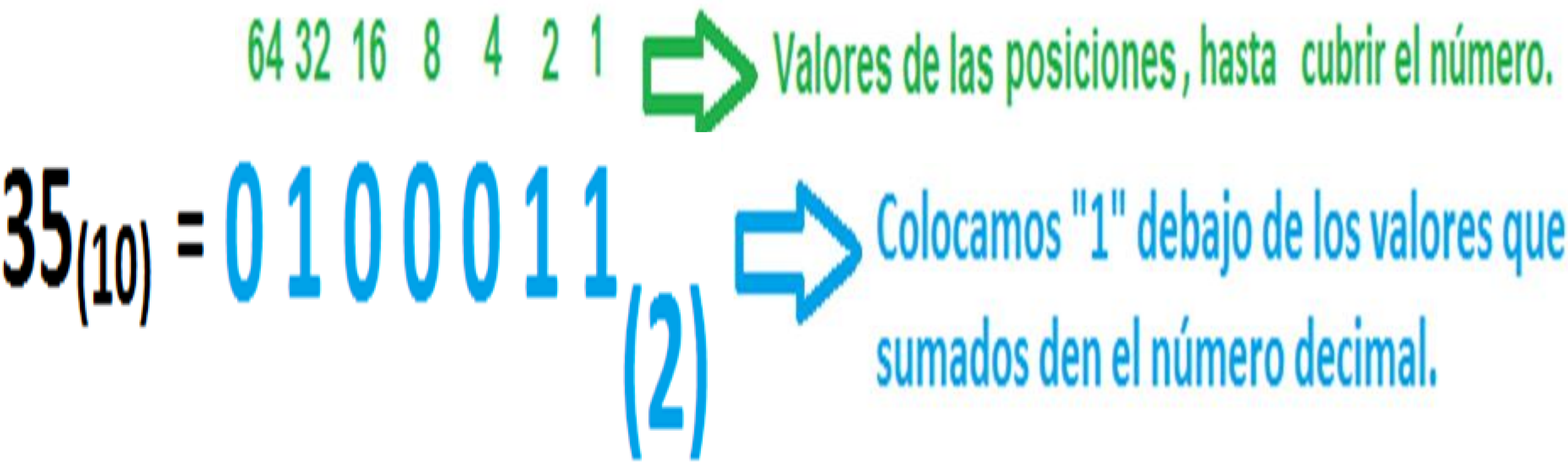
Sumamos los valores donde están los 1

$$32 + 8 + 4 + 2 + 1 = 47_{(10)}$$

Cambios de base

Ejercicio:

Dado el número decimal, pasarlo a **base 2**:



El número binario debe comenzar con un cero “0” ya que indica que es un número positivo (bit de signo).

Cambios de base

El Sistema Binario se relaciona con el Sistema Hexadecimal (base 16).

¿Porqué grupos de 4 dígitos?

Cambios de base

Ejercicio:

Dado el número binario **01101111**$_{(2)}$, pasarlo a **base 16**:

Separar el número binario en grupos de 4 dígitos, de derecha a izquierda.

En cada grupo colocar el valor de las posiciones.

0110	1111$_{(2)}$	
8 4 2 1	8 4 2 1	⇐ Valores de las posiciones

4 + 2 = 6	8 + 4 + 2+ 1 = 15 = F

En cada grupo sumamos los valores donde están los 1.

El resultado es:

6F$_{(16)}$

Cambios de base

Ejercicio:

Dado el número hexadecimal pasarlo a **base 2**:

$4B_{(16)}$

A cada dígito hexadecimal escribirlo con 4 dígitos binarios.

4	B
8 4 2 1	8 4 2 1
0100	1011

Valores de las posiciones

Debajo de cada valor ponemos unos, tal que sumados den el dígito hexadecimal. El resto se completa con ceros.

El resultado es: $01001011_{(2)}$

Cambios de base

El Sistema Decimal <u>NO</u> se relaciona con el Sistema Hexadecimal, sí con el Binario.

Cambios de base

Ejercicio:

Dado el número hexadecimal **4B**$_{\mathbf{(16)}}$, pasarlo a **base 10**:

$$\overbrace{8\ 4\ 2\ 1}^{4}\ \ \overbrace{8\ 4\ 2\ 1}^{B}$$

$4B_{(16)} = 0100\ 1011_{(2)}$ Primero se pasa a binario.

Luego a Decimal

128 64 32 16 8 4 2 1 Valores de todas las posiciones

Sumamos los valores donde están los 1

$64 + 8 + 2 + 1 = 75_{(10)}$

Cambios de base

Ejercicio:

Dado el número decimal pasarlo a **base 16**:

Primero se pasa a binario.

64 32 16 8 4 2 1

Valores de las posiciones, hasta cubrir el número.

$$35_{(10)} = 00100011_{(2)}$$

| 8 4 2 1 | 8 4 2 1 |

Separar el número binario en grupos de 4 dígitos, de derecha a izquierda.
En cada grupo colocar el valor de las posiciones.

| 2 | 2+ 1 = 3 |

En cada grupo sumamos los valores donde están los 1.

El resultado es: $23_{(16)}$

Cambios de base

De CUALQUIER SNP a DECIMAL:

DÌGITO X (BASE DE ORIGEN)$^{\text{POSICIÒN}}$ +
DÌGITO X (BASE DE ORIGEN)$^{\text{POSICIÒN}}$ +

….

= RESULTADO EN BASE 10

Cambios de base

Ejercicio:

Dado el número hexadecimal pasarlo a base 10:

1 0

Posiciones, comienzan en 0.
Van de derecha a izquierda.

$$2D_{(16)} =$$

MAL SI PONEN $213_{(10)}$

$$2 * (16)^1 + D * (16)^0 =$$

$$2 * (16)^1 + 13 * (16)^0 =$$

$$32 + 13 =$$

$$45_{(10)}$$

SUMAS

Recordemos el proceso de sumar en decimal:

$$\begin{array}{r} {\scriptstyle 1\,1\;\;} \\ 297 \\ +\;\;446 \\ \hline 743 \end{array}$$

$7 + 6 = 13$

El resultado “13” es mayor que la base, entonces se debe restar la base.

$13 - 10 = 3$

Ahora “3” es menor que la base. Pongo “3” y me “llevo” las veces que se resto la base.

Suma en binario:

+	0	1
0	0	1
1	1	10

Podemos usar esta tabla o relacionar como se suma en decimal.

```
      1 1
  0 0 1 1 0 1
+ 0 0 1 1 1 0
  -----------
  0 1 1 0 1 1
```

1 + 1 = 2 - 2 = 0

"Llevo" un 1, porque reste la base una vez.

1 + 1 + 1 = 3 - 2 = 1

"Llevo" un 1

Se pone un "0" como bit de signo para indicar que es un nùmero binario positivo.

Tanto para la suma como para la resta de números binarios, estos deben tener la <u>misma cantidad de dígitos</u>.
Si no tienen, se completan con ceros a la izquierda hasta llegar a la misma cantidad de dígitos.

Suma en hexadecimal:

Sumamos relacionando como se suma en decimal.

		1		
	A	3	F	15 + 12 = 27 - 16 = 11 = B "Llevo" un 1.
+	4	B	C	1 + 3 + 11 = 15 = F
	E	F	B	10 + 4 = 14 = E

RESTAS

Recordemos el proceso de restar en decimal:

$$\begin{array}{rr} & \quad 10 \\ & 1\;+ \\ & \cancel{2}6 \\ - & 8 \\ \hline & 18 \end{array}$$

6 – 8 NO SE PUEDE,
entonces "PIDO" LA BASE, quedando:

6 + 10 = 16 → ahora sí se puede restar

16 – 8 = 8

El que "PRESTA" LA BASE queda con uno menos.

Resta en hexadecimal:

Restamos relacionando como se resta en decimal.

$$\begin{array}{rccc} & & & 16 \\ & & E & + \\ & E & \cancel{F} & B \\ - & A & 3 & F \\ \hline & 4 & B & C \end{array}$$

B - F = 11 - 15 → NO se puede restar

11 + 16 → PIDO LA BASE

11 + 16 = 27 → ahora sí se puede restar

27 – F = 27 – 15 = 12 = C

El que "PRESTA" LA BASE queda con uno menos.

E – 3 = 14 – 3 = 11 = B

E – A = 14 – 10 = 4

Resta en Binario:

En binario ambos números deben tener igual cantidad de dígitos.

$$\begin{array}{rcccc} & & & 1 & 2 \\ & & 0 & \cancel{2} & + \\ & 0 & \cancel{1} & \cancel{0} & 0 \\ - & 0 & 0 & 0 & 1 \\ \hline & 0 & 0 & 1 & 1 \end{array}$$

0 - 1 → NO se puede restar

PIDO LA BASE

SI EL DÍGITO DE AL LADO NO PUEDE PRESTAR PEDIMOS AL SIGUIENTE QUE PUEDA PRESTAR.

El que "PRESTA" LA BASE queda con uno menos.

AHORA SÍ PUEDO RESTAR

0 + 2 = 2

2 – 1 = 1

1 – 0 = 1

0 – 0 = 0

La Unidad Aritmético Lógica (ALU)
NO RESTA.

Para poder restar
<u>complementa</u> al <u>número negativo</u>
y suma,
pero internamente esta restando.

RESTAS USANDO COMPLEMENTO

Siempre hay que partir de un número binario positivo bien escrito, con bit de signo 0.

Complemento a la base menos 1 C1

Consiste en cambiar 0 por 1 y 1 por 0.

	4	2	1	
$+3_{(10)} =$	0	1	1	$-3_{(10)} =$ 1 0 0
	+			-

	8	4	2	1	
$+6_{(10)} =$	0	1	1	0	$-6_{(10)} =$ 1 0 0 1
	+				-

Complemento a la base o C2

Copiar los dígitos binarios de derecha a izquierda hasta que encuentre el primer 1 inclusive, luego cambiar 0 por 1 y 1 por 0.

4 2 1

$+3_{(10)} =$ 0 1 1 $-3_{(10)} =$ 1 0 1

\+ -

8 4 2 1

$+6_{(10)} =$ 0 1 1 0 $-6_{(10)} =$ 1 0 1 0

\+ -

EJEMPLO:

Realizar las operaciones

$A + B$

$A + (-B)$

$(-A) + B$

$(-A) + (-B)$

En ambos complementos

$A = 70$

$B = 80$

*Se debe comenzar pasando de decimal a binario, con **bit de signo 0**, indicando que parten de un número decimal positivo.*

*Para sumar, ambos números deben tener **igual cantidad de dígitos,** si no tienen se completa con 0 a la izquierda.*

	128	64	32	16	8	4	2	1
+ 70	0	1	0	0	0	1	1	0
+ 80	0	1	0	1	0	0	0	0

Luego pasamos a binarios negativos, mediante complementos.

	128	*64*	*32*	*16*	*8*	*4*	*2*	*1*		**En C1**	**En C2**
+ 70	0	1	0	0	0	1	1	0	- 70	1 0 1 1 1 0 0 1	1 0 1 1 1 0 1 0
+ 80	0	1	0	1	0	0	0	0	- 80	1 0 1 0 1 1 1 1	1 0 1 1 0 0 0 0

	128 64 32 16 8 4 2 1		En C1	En C2
+ 70	0 1 0 0 0 1 1 0	- 70	1 0 1 1 1 0 0 1	1 0 1 1 1 0 1 0
+ 80	0 1 0 1 0 0 0 0	- 80	1 0 1 0 1 1 1 1	1 0 1 1 0 0 0 0

A + B = 70 + 80 = 150

Trabajamos con los dos números positivos.

70 + 80 = 150
01000110 + 01010000 10010110

En rojo se resalta el ***OVERFLOW***
(positivo + positivo da negativo)

SE RESUELVE AGREGANDO NUEVO BIT DE SIGNO, 0.

Corrección, con nuevo bit de signo
001000110 + 001010000 010010110 = + 150

RESULTADO CORRECTO →

Trabajamos con los números corregidos.

	256 128 64 32 16 8 4 2 1		***En C1***	***En C2***
+ 70	0 0 1 0 0 0 1 1 0	- 70	1 1 0 1 1 1 0 0 1	1 1 0 1 1 1 0 1 0
+ 80	0 0 1 0 1 0 0 0 0	- 80	1 1 0 1 0 1 1 1 1	1 1 0 1 1 0 0 0 0

$$(-A) + (-B) = (-70) + (-80) = -150$$

C1:

Último acarreo en C1 se lo vuelve a sumar.

```
1    1  1 1 1 1 11
       110111001
    +  110101111
    ------------
       101101000

   +           1
   -------------
       101101001
```

El resultado esta expresado en complemento a 1, para saber que valor es lo "**descomplementamos**", sacamos el binario positivo, lo pasamos a decimal y sabremos el resultado, recordando que es un número negativo.

101101001
a C1:
010010110 en decimal es 150

RECORDEMOS QUE VIENE DE UN NEGATIVO, ENTONCES EL RESULTADO ES - 150

Trabajamos con los números corregidos.

	256 128 64 32 16 8 4 2 1		En C1	En C2
+ 70	0 0 1 0 0 0 1 1 0	- 70	1 1 0 1 1 1 0 0 1	1 1 0 1 1 1 0 1 0
+ 80	0 0 1 0 1 0 0 0 0	- 80	1 1 0 1 0 1 1 1 1	1 1 0 1 1 0 0 0 0

$$(-A) + (-B) = (-70) + (-80) = -150$$

C2:

$$
\begin{array}{r}
1\ \ 1\,1 \\
110111010 \\
+\ \ 110110000 \\
\hline
\cancel{1}\ 101101010
\end{array}
$$

En C2
el último acarreo se descarta.

El resultado esta expresado en complemento a 2, para saber que valor es lo "**descomplementamos**", sacamos el binario positivo, lo pasamos a decimal y sabremos el resultado, recordando que es un número negativo.

101101010
a C2:
010010110 es en decimal 150.

RECORDEMOS QUE VIENE DE UN NEGATIVO, ENTONCES EL RESULTADO ES - 150

	256	128	64	32	16	8	4	2	1		En C1	En C2
+ 70	0	0	1	0	0	0	1	1	0	- 70	1 1 0 1 1 1 0 0 1	1 1 0 1 1 1 0 1 0
+ 80	0	0	1	0	1	0	0	0	0	- 80	1 1 0 1 0 1 1 1 1	1 1 0 1 1 0 0 0 0

$$A + (-B) = 70 + (-80) = -10$$

C1:

$$\begin{array}{r} 001000110 \\ +\ 110101111 \\ \hline 111110101 \end{array}$$

C2:

$$\begin{array}{r} 001000110 \\ +\ 110110000 \\ \hline 111110110 \end{array}$$

En ambos casos se "**descomplementa** el número negativo para saber cual es el positivo original. Luego se pasa a decimal.

000001010 =
10 en decimal

EL RESULTADO EN
DECIMAL SERÁ -10

	256 128 64 32 16 8 4 2 1		En C1	En C2
+ 70	0 0 1 0 0 0 1 1 0	- 70	1 1 0 1 1 1 0 0 1	1 1 0 1 1 1 0 1 0
+ 80	0 0 1 0 1 0 0 0 0	- 80	1 1 0 1 0 1 1 1 1	1 1 0 1 1 0 0 0 0

$$(-A) + B = (-70) + 80 = 10$$

C1:

Último acarreo en C1 se lo vuelve a sumar.

```
 1 1111
  110111001
+ 001010000
-----------
  000001001
+         1
-----------
  000001010
```

C2:

```
   110111010
+  001010000
------------
~~1~~ 000001010
```

En C2 el último acarreo se descarta.

EL RESULTADO EN DECIMAL SERÁ 10

MULTIPLICACIONES

Multiplicación en binario

La tabla de multiplicar para números binarios es la siguiente:

x	0	1
0	0	0
1	0	1

$$\begin{array}{r} 6 \\ \times\ 5 \\ \hline 30 \end{array}$$

$$\begin{array}{r} 0110 \\ \times\ 0101 \\ \hline 0110 \\ 0000 \\ 0110 \\ 0000 \\ \hline 0011110 \end{array}$$

OTRAS FORMAS DE CODIFICACIÒN

CÒDIGO
DE
TÈTRADAS
BCD

BCD

- Representa ***<u>solo</u>*** *dígitos decimales* (0 al 9).

- Las tétradas no usadas se llaman pseudotetradas.

BCD

TÉTRADAS	BCD (8421)
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	Pseudotétradas
1011	Pseudotétradas
1100	Pseudotétradas
1101	Pseudotétradas
1110	Pseudotétradas
1111	Pseudotétradas

BCD

<u>EJEMPLO</u>: pase el número decimal 130 a BCD.

1	3	0
8 4 2 1	8 4 2 1	8 4 2 1
0 0 0 1	0 0 1 1	0 0 0 0

$$130_{(10)} = 0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0_{\textbf{(BCD)}}$$

BCD ≠ BINARIO

$130_{(10)} =$

1	3	0
8 4 2 1	8 4 2 1	8 4 2 1

0 0 0 1 0 0 1 1 0 0 0 0 $_{(BCD)}$

256	128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	1	0

$_{(2)}$

CÒDIGO

ASCII

ASCII

- Representa *caracteres alfanuméricos, imprimibles y no imprimibles.*

- Usa 7 bits para representar un carácter:

$$B_6 \quad B_5 \quad B_4 \quad B_3 \quad B_2 \quad B_1 \quad B_0$$

ASCII

$B_6 B_5 B_4$								
000	**001**	**010**	**011**	**100**	**101**	**110**	**111**	**$B_3B_2B_1B_0$**
NUL	DLE	SP	0	@	P	,	p	**0000**
SOH	DC1	!	1	A	Q	a	q	**0001**
STX	DC2	“	2	B	R	b	r	**0010**
ETX	DC3	#	3	C	S	c	s	**0011**
EOT	DC4	$	4	D	T	d	t	**0100**
ENQ	NAK	%	5	E	U	e	u	**0101**
ACK	SYN	&	6	F	V	f	v	**0110**
BEL	ETB	‘	7	G	W	g	w	**0111**
BS	CAN	(	8	H	X	h	x	**1000**
HT	EM	)	9	I	Y	i	y	**1001**
LF	SUB	*	:	J	Z	j	z	**1010**
VT	ESC	+	;	K	[	k	{	**1011**
FF	FS	´	<	L	\	l	¦	**1100**
CR	GS	-	=	M	]	m	}	**1101**
SO	RS	.	>	N	^	n	~	**1110**
ST	US	/	?	O	_	o	DEL	**1111**

ASCII

EJEMPLO: pase el número decimal 130 a ASCII.

1							3							0						
B_6	B_5	B_4	B_3	B_2	B_1	B_0	B_6	B_5	B_4	B_3	B_2	B_1	B_0	B_6	B_5	B_4	B_3	B_2	B_1	B_0
0	1	1	0	0	0	1	0	1	1	0	0	1	1	0	1	1	0	0	0	0

$130_{(10)} =$

0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 0 0 (ASCII)

ASCII

EJEMPLO: pase **Ana** a código ASCII.

A							n							a						
B_6	B_5	B_4	B_3	B_2	B_1	B_0	B_6	B_5	B_4	B_3	B_2	B_1	B_0	B_6	B_5	B_4	B_3	B_2	B_1	B_0
1	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**

(ASCII)