Universidad Autónoma del Estado de México
Centro Universitario UAEM Valle de México

Centro Universitario
UAEM Valle de México

Ingeniería en Computación

Unidad de Aprendizaje: Lenguaje Ensamblador

Tema: Sistemas Numéricos

Elaboró: Dr. en C. Héctor Rafael Orozco Aguirre
Octubre de 2016

PROGRAMA DE ESTUDIO POR COMPETENCIAS
LENGUAJE ENSAMBLADOR

I. IDENTIFICACIÓN DEL CURSO

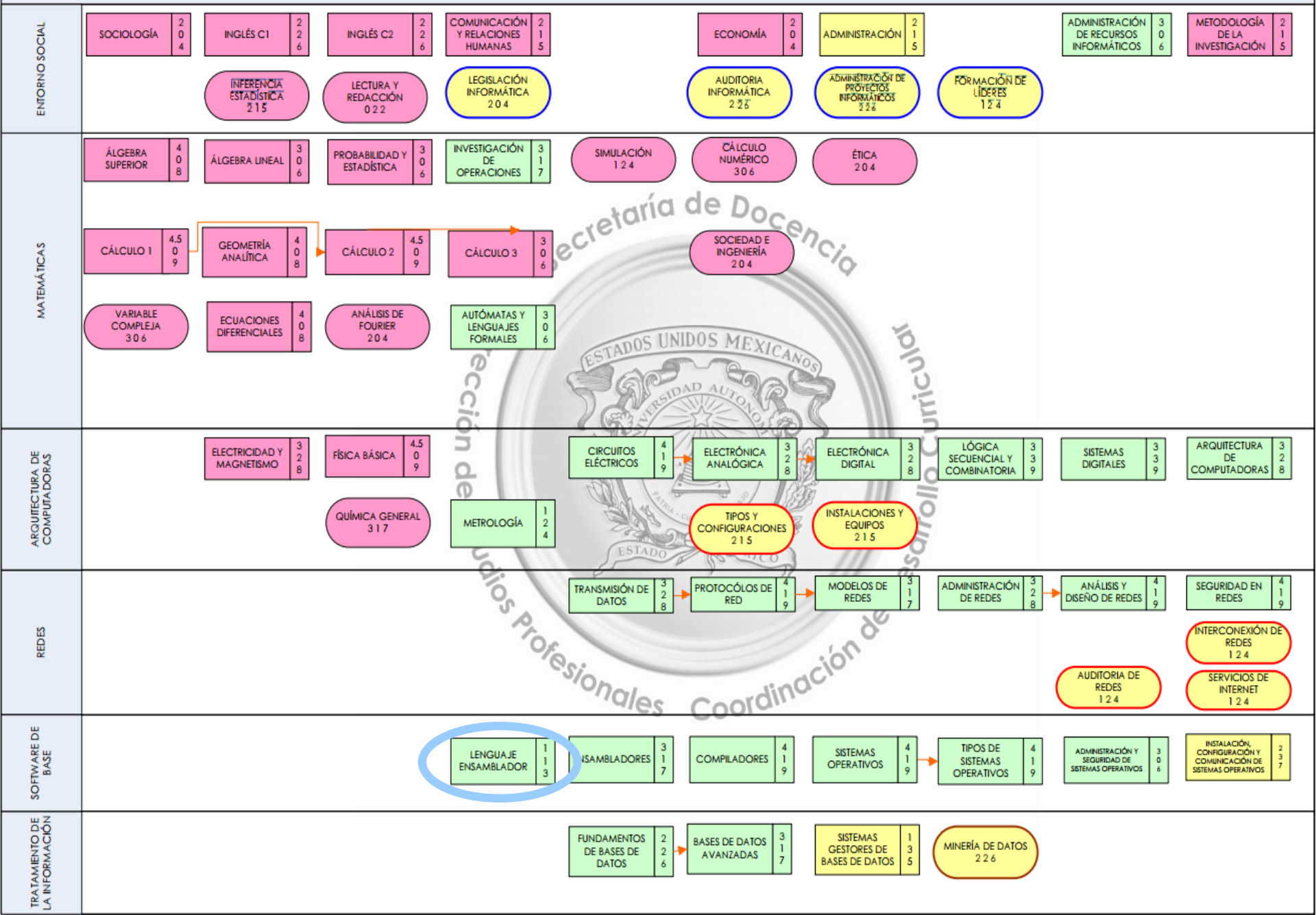
<table>
<tr><td colspan="7">Espacio Educativo: Facultad de Ingeniería</td></tr>
<tr><td colspan="3">Licenciatura: Ingeniería en Computación

Año de aprobación por el Consejo Universitario:</td><td colspan="4">Área de docencia: Software de base</td></tr>
<tr><td colspan="2" rowspan="2">Aprobación por los H.H. Consejos Académico y de Gobierno</td><td rowspan="2">Fecha:</td><td colspan="2">Programa elaborado por:
Benjamín Pérez Clavel,
Felipe Camacho M.</td><td colspan="2">Programa revisado por:

Miembros de la academia</td></tr>
<tr><td colspan="4">Fecha de elaboración :
Septiembre 2009
Actualización:
Noviembre 2013</td></tr>
<tr><td>Clave</td><td>Horas de teoría</td><td>Horas de práctica</td><td>Total de horas</td><td>Créditos</td><td>Tipo de curso</td><td>Núcleo de formación</td></tr>
<tr><td>L41047</td><td>1</td><td>1</td><td>2</td><td>3</td><td>Curso - Taller</td><td>Sustantivo</td></tr>
<tr><td colspan="4">Unidad de Aprendizaje Antecedente
Ninguna</td><td colspan="3">Unidad de Aprendizaje Consecuente
Ninguna</td></tr>
</table>

MAPA CURRICULAR DE LA LICENCIATURA DE INGENIERÍA EN COMPUTACIÓN 2004

CRÉDITOS TOTALES: 430-450



Propósito de la Unidad de Aprendizaje

- Programar en lenguaje ensamblador aplicaciones de software o hardware para tener el control total de un sistema de cómputo utilizando para dicho aprendizaje un equipo de cómputo (PC) o un microcontrolador/microprocesador comercial.

- El alumno desarrollará programas en lenguaje ensamblador de uso práctico para manejar los componentes básicos de un sistema de cómputo, usando las instrucciones y las metodologías propias del la estructura del lenguaje ensamblador.

- El alumno deberá realizar, explicar, documentar cada programa realizado, de tal forma que realce la comprensión de las instrucciones individuales y el estilo de programación

Contenido

- Organización de datos
- Sistemas numéricos
- Representación de valores
- Conversiones entre sistemas o bases
- Operaciones aritméticas en las bases
- Complementos a 1 y a 2

Guion explicativo

- Esta presentación tiene como fin dar a conocer a los alumnos los siguientes aspectos:
  - ¿Cuáles son los sistemas numéricos que hay?
  - ¿Qué es un sistema posicional y no posicional?
  - Tipos de notaciones para representación de valores
  - Técnicas empleadas para cambios entre bases
  - ¿Cómo realizar operaciones aritméticas en las bases?
  - Representaciones en complemento a 1 y a 2 entre las bases

Guion explicativo

- El contenido de esta presentación contiene temas de interés contenidos en la Unidad de Aprendizaje de Lenguaje Ensamblador.

- Las diapositivas deben explicarse en orden, y deben revisarse aproximadamente en 6 horas, además de realizar preguntas y dejar ejercicios a la clase sobre el contenido mostrado.

Organización de datos

- Las computadoras comprenden el lenguaje de los números.
- La organización de una computadora depende entre otros factores del sistema de representación numérica adoptado.
- Se trabaja con el sistema binario, de donde proviene el término bit como contracción de “**bi**nary digi**t** o **b**inary dig**it**”.

Sistemas numéricos

- Los ***sistemas de numeración*** *o* ***sistemas numéricos*** son un conjunto de símbolos que se utilizan para representar cantidades según ciertas reglas.

- Números como cantidades:
  - reales negativos
  - reales positivos
  - enteros negativos
  - enteros positivos

Sistemas numéricos

- Un sistema de numeración puede representarse como ***N = S + R*** donde:
  - **N** es el sistema de numeración considerado
  - **S** son los símbolos permitidos en el sistema.
  - **R** son las reglas de generación que nos indican qué números son válidos y cuáles son no-válidos en el sistema.

Sistemas numéricos

- ***Sistema decimal*:** Es el sistema de numeración utilizado en la vida cotidiana, cuya base es diez, utilizando los símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

- ***Sistema binario***: los dos símbolos utilizados son el 0 y el 1, los que reciben el nombre de bit.

- ***Sistema octal*:** de base 8, los símbolos utilizados son 0, 1, 2, 3, 4, 5, 6,7.

- ***Sistema hexadecimal***: de base 16, los símbolos utilizados son 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Sistemas numéricos

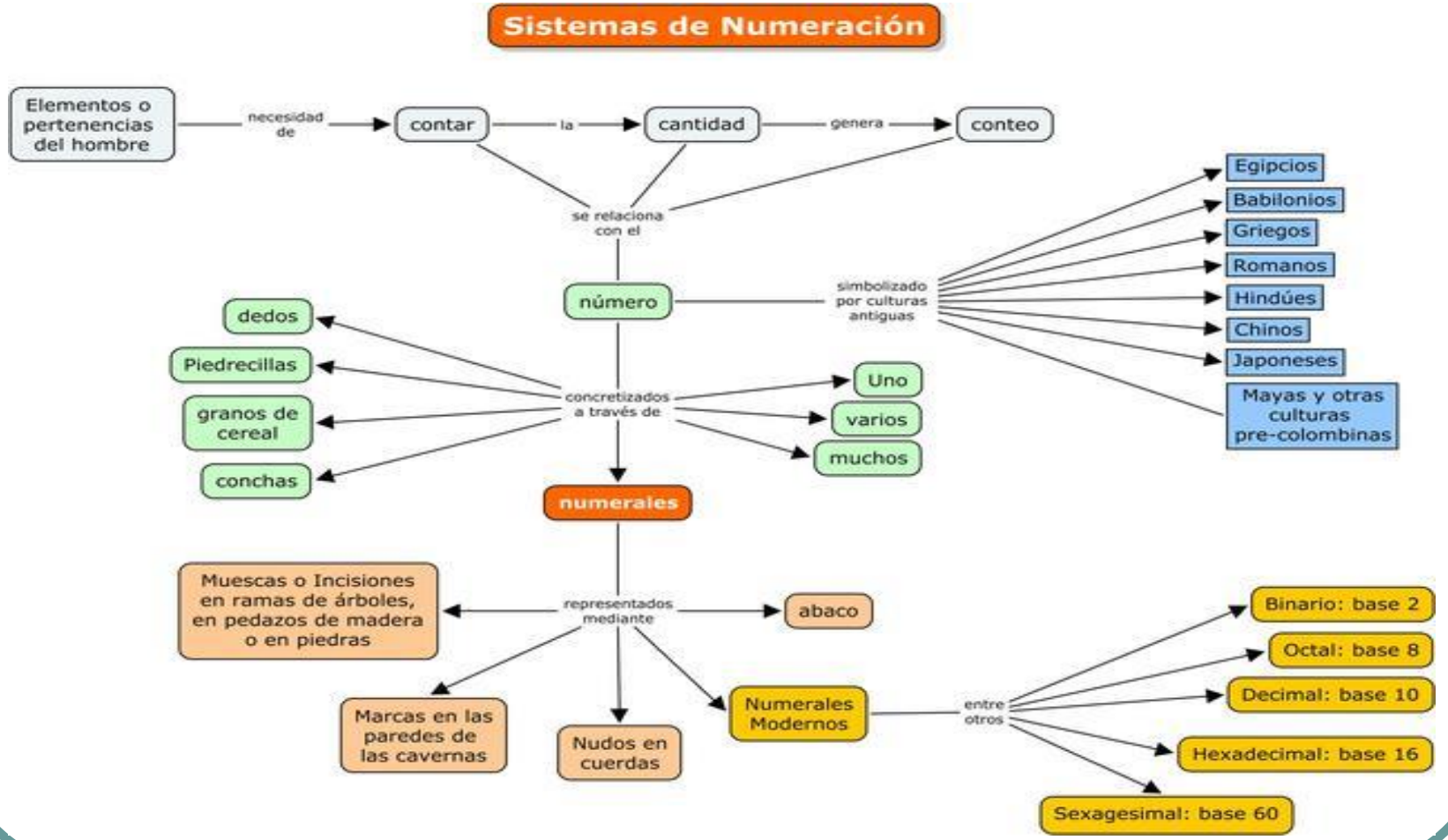


Imagen tomada de: http://estudiandoconangela.weebly.com/sistema-de-numeracioacuten.html

Sistemas numéricos

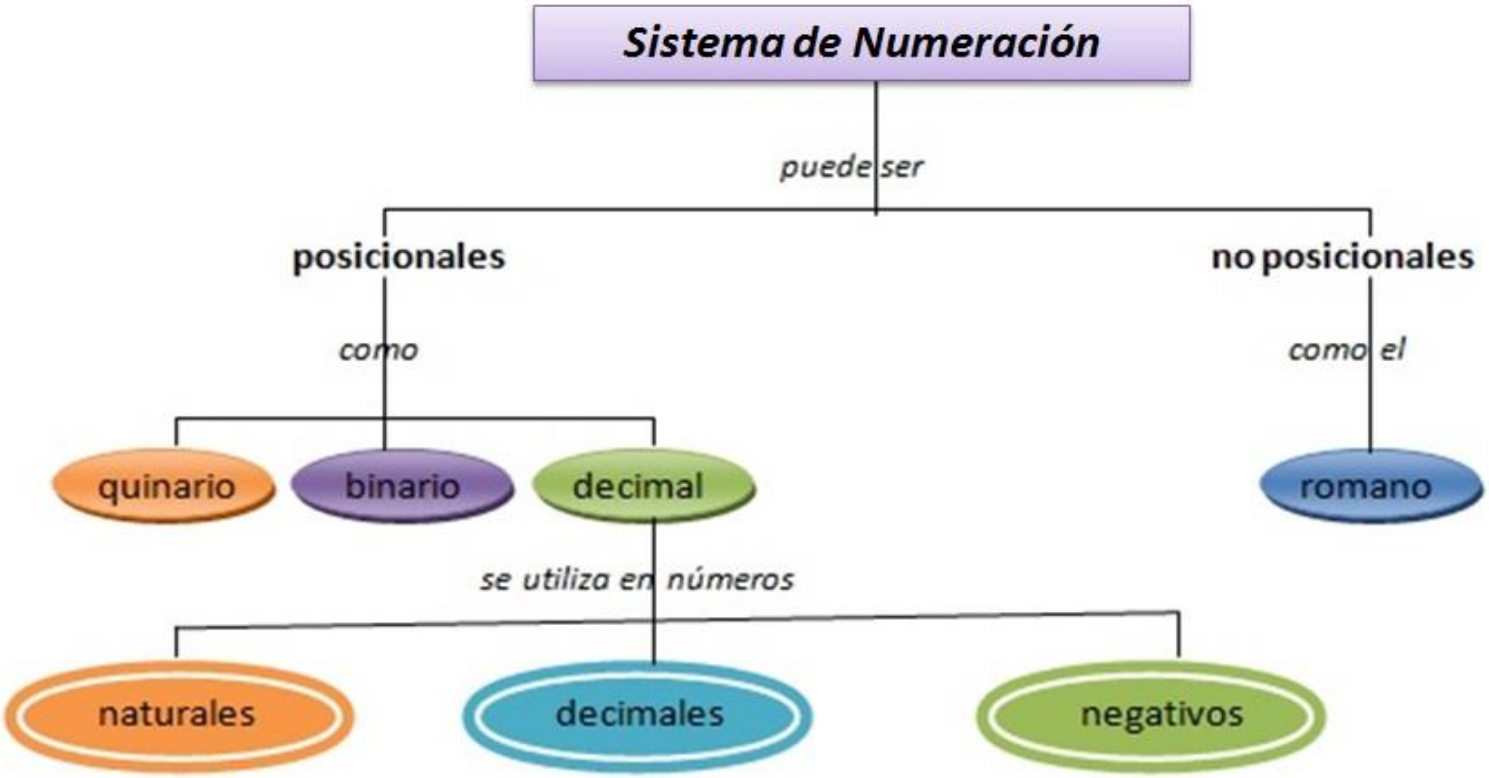


Imagen tomada de: http://estudiandoconangela.weebly.com/sistema-de-numeracioacuten.html

Sistema no posicional

- **Sistema de numeración no posicional:** cada cifra siempre tiene el mismo valor, independientemente del lugar que éstas ocupen.
- Los números romanos son un claro ejemplo de ello.
  - LX = 50 + 10 = 60
  - XL = 50 – 10 = 40
  - VI = 5 + 1 = 6
  - IV = 5 – 1 = 4

Sistema posicional

- **Sistema de numeración posicional:** cada cifra tiene un valor propio y un valor según la posición en la que se encuentra.

- El sistema decimal, el binario, el octal y el hexadecimal son los más usados.
  - $0111_2 = 7_{10}$
  - $0111_2 = 7_8$
  - $0111_2 = 7_{16}$

Teorema fundamental de los números

- La representación de una cantidad en un sistema de numeración distinto a decimal, es dado por la formula:

$$N = \sum_{i=-d}^{n} X_i x B^i$$

- B = base del sistema de numeración.
- i = posición respecto al punto.
- d = número de cifras a la derecha.
- n = Numero de cifras a la izquierda del numero menos 1.
- X = Cada una de las cifras que componen el número.

Representación de un valor

- Se toma un número **b**, base del sistema de numeración y todo número N se representa como la combinación de potencias de aquel coeficiente que toman valores de 0 a b-1, en la forma:

$$a_k b^k + a_{k-1} b^{k-1} + \ldots + a_1 p + a_0$$

Representación de un valor

- En un sistema de numeración posicional de base b, la representación de un número se define a partir de la regla:

$$(\ldots a_3 a_2 a_1 a_0 . a_{-1}\ a_{-2}\ a_{-3}\ \ldots)_b = \ldots + a_2 b^2 + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + a_{-2} b^{-2} + a_{-3} b^{-3} + \ldots$$

  - ***b*** es un entero no negativo mayor a 1 y cuando los a_i pertenecen al conjunto de enteros en el rango $0 \leq a_i < b$
  - El punto que aparece entre los dígitos a_0 y a_{-1} se denomina punto fraccionario.
  - Cuando ***b*** = 10 se le llama punto decimal y cuando *b* = 2, punto binario.

Representación de un valor

- Entonces este número se denota abreviadamente:

$$(a_k a_{k-1} \ldots a_1 a_0)_b$$

- Este caso expresa que el valor de cada cifra depende del lugar que ocupa.

Representación yuxtaposicional

- Los dígitos más a la izquierda y más a la derecha tienen una valor significativo.
  - El que se encuentra más a la izquierda se denomina como el dígito más significativo (MSD).
  - El que se encuentra más a la derecha se denomina como el dígito menos significativo (LSD).

Representación yuxtaposicional

- $(195950.715)_{10}$
  - Base del sistema numérico b = 10.
  - Parte entera n = 6 dígitos.
  - Parte decimal m = 3 dígitos.

Representación polinomial

- Cualquier número N puede ser escrito como un polinomio en potencias de la base.

- 424.59_{10}
- $N = 424.59_{10}$
  $= 4*10^2 + 2*10^1 + 4*10^0 + 5*10^1 + 9*10^{-2}$

Conversiones ente sistemas o bases

- Sea el número $a_k a_{k-1} \ldots a_1$, un entero en base R. Para convertir este número de base R a base Q se utiliza la conversión:

$$a_k R^{k-1} + a_{k-1} R^{k-2} + \ldots + a_1 R^0$$

  - R es la base en la que se encuentra el número (base actual), k es el número de dígitos que conforman el número y Q es la nueva base (se debe trabajar con aritmética en base Q).

Ejemplos:

1) Convertir $(100110)_2 \rightarrow (\quad)_{10}$

R=2 k=6 Q=10 a6=1, a5=0, a4=0, a3=1, a2=1, a1=0

$1*2^{6-1}+0*2^{6-2}+0*2^{6-3}+1*2^{6-4}+1*2^{6-5}+0*2^{6-6} = 1*2^5+0*2^4+0*2^3+1*2^2+1*2^1+0*2^0 =$
$1*2^5+1*2^2+1*2^1 = 32+4+2=38$

Finalmente obtenemos que: $(100110)_2 \rightarrow (38)_{10}$

Conversiones ente sistemas o bases

Convertir $100110_2 \rightarrow ?_{10}$

R=2 k=6 Q=10 a6=1, a5=0, a4=0, a3=1, a2=1, a1=0

$1*2^{6-1} + 0*2^{6-2} + 0*2^{6-3} + 1*2^{6-4} + 1*2^{6-5} + 0*2^{6-6} =$

$1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 = 1*2^5 + 1*2^2 + 1*2^1 = 32 + 4 + 2 = 38$

Por lo tanto: $100110_2 \rightarrow 38_{10}$

Conversiones ente sistemas o bases

- Ejercicios en clase:
  - $111111100110_2 \rightarrow ?_{10}$
  - $101010000110_2 \rightarrow ?_{10}$
  - $100011100110_2 \rightarrow ?_{10}$
  - $100110011110_2 \rightarrow ?_{10}$

Conversiones ente sistemas o bases

Convertir $4302_5 \rightarrow ?_3$

R=5 Q=3 k=4 a4=4, a3=3, a2=0, a1=2

Se debe trabajar con aritmética en base 3, por ende, se necesitan las tablas de suma y multiplicación en base 3.

+	0	1	2
0	0	1	2
1	1	2	10
2	2	10	11

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	11

Conversiones ente sistemas o bases

$4302_5 = 4*5^3 + 3*5^2 + 0*5^1 + 2*5^0 = 11*12^3 + 10*12^2 + 2*12^0$

$= 11*11122+10*221+2*1$

$= 200112 + 2210 + 2 = 210101$

12*12	221*12	11122*11	221*10	200112
101	1212	11122	000	+ 2210
12	221	11122	221	2
221	11122	200112	2210	210101

Se tiene por consiguiente que: $4302_5 \rightarrow 210101_3$

Conversiones ente sistemas o bases

- Ejercicios en clase:
  - $3420_5 \rightarrow ?_3$
  - $32_5 \rightarrow ?_3$
  - $420_5 \rightarrow ?_3$
  - $30_5 \rightarrow ?_3$

Conversión de base X a base 10 (X ≠ **10**)

Algoritmo A.

1. i←k, num ←0
2. Mientras i≥1 hacer
   num ←num*R+a_i
   i ←i-1
   fin_mientras
3. Fin

Donde R es la base actual, k es el número de dígitos que componen el número y a_i es el i-ésimo dígito del número en base X (derecha a izquierda)

Convertir 4302_5 → $?_{10}$

I	num	R	a4	a3	a2	a1	k
4	0	5	4	3	0	2	4
3	4						
2	23						
1	115						
0	577						

Por lo tanto: 4302_5 → 577_{10}

Conversión de base 10 a base S (S ≠ **10**)

Algoritmo B.

1. i←1, q←0, p ←0
2. Repetir
   q ←[x/s] (parte entera)
   p ←x-q*s (residuo)
   a_i ←p, i ←i+1, x ← q
   hasta q=0
3. Fin

Convertir 577_{10} → $?_3$

Se concluye que: 577_{10} → 210101_3

Donde x inicialmente es el número a convertir, s es la nueva base y a_i es el i-ésimo dígito del número en base s tomando el orden $a_k a_{k-1} \ldots a_1$

x	q	p	ai	i	s
577	0	0		1	3
192	192	1	1	2	
64	64	0	0	3	
21	21	1	1	4	
7	7	0	0	5	
2	2	1	1	6	
0	0	2	2	7	

Conversión de base X a base 10 (X ≠ **10**)

Números fraccionarios

Algoritmo C.

1. i←m, num←0
2. Mientras i≥1hacer
   num ←(num+bi)/R
   i ←i-1
   fin_mientras
3. Fin

Donde m es el número de dígitos que componen el número que queremos convertir, R es la base actual y num es el número en la nueva base.

Convertir $.A06_{16}$ → $?_{10}$

i	num	m	R	b1	b2	b3
3	0	3	16	A	0	6
2	.375					
1	.0234375					
0	.62646484					

Por lo tanto: $.A06_{16}$ → $.62646484_{10}$

Conversión de base 10 a base S (S ≠ **10**)

Números fraccionarios

Algoritmo D.

1. i←1
2. Mientras i ≤ m hacer
   x ←x*s
   y ←[x] (parte entera)
   x ←x-y, bi ←y, i ← i+1
   fin_mientras
3. Fin

Convertir $.62646484_{10}$ → $?_7$

Se tiene que: $.62646484_{10}$ → $.424_7$

Donde m es el número de dígitos que se desean obtener, x es el número a convertir inicialmente, s es la nueva base y b_i es el i-ésimo dígito del número en base s tomando el orden $b_1b_2\ldots b_m$

i	x	y	bi	m	s
1	.62646484			3	7
	4.38525388	4			
2	.38525388		4		
	2.69677716	2			
3	.69677716		2		
	4.87744012	4			
4	.87744012		4		

Método de restas sucesivas

Represar 104_{10} en base 3

$$104 - 81 = 23 \text{ , } 81 = 3^4 \times 1$$
$$23 - 0 = 23 \text{ , } 0 = 3^3 \times 0$$
$$23 - 18 = 5 \text{ , } 18 = 3^2 \times 2$$
$$5 - 3 = 2 \text{ , } 3 = 3^1 \times 1$$
$$2 - 0 = 2 \text{ , } 0 = 3^0 \times 2$$

$$104_{10} = 10212_3$$

Método de restas sucesivas

- Ejercicios en clase:
  - $13420_{10} \rightarrow ?_{3}$
  - $532_{10} \rightarrow ?_{8}$
  - $420_{10} \rightarrow ?_{2}$
  - $730_{10} \rightarrow ?_{16}$

Método del resto de cocientes

Represar 104_{10} en base 3

104 % 3 = 2 , 104 / 3 = 34
34 % 3 = 1 , 34 / 3 = 11
11 % 3 = 2 , 11 / 3 = 3
3 % 3 = 0 , 3 / 3 = 1
1 % 3 = 1 , 1 / 3 = 0

$104_{10} = 10212_3$

Método del resto de cocientes

Mediante la siguiente tabla se puede entender mejor cómo represar 104_{10} en base 3

Dividendo	Divisor	Cociente	Residuo
104	3	34	2
34	3	11	1
11	3	3	2
3	3	1	0
1	3	0	1

Leídos de abajo hacia arriba

$$104_{10} = 10212_3$$

Escritos de izquierda a derecha

Método del resto de cocientes

- Ejercicios en clase:
  - $13420_{10} \rightarrow ?_{3}$
  - $532_{10} \rightarrow ?_{8}$
  - $420_{10} \rightarrow ?_{2}$
  - $730_{10} \rightarrow ?_{16}$

Método de restas sucesivas para fracciones

Represar 0.4304_{10} en base 5

$0.4304 - 0.4000 = 0.0304 , 0.4000 = 5^{-1} \times 2$
$0.0304 - 0.0000 = 0.0304 , 0.0000 = 5^{-2} \times 0$
$0.0304 - 0.0240 = 0.0064 , 0.0240 = 5^{-3} \times 3$
$0.0064 - 0.0064 = 0.0000 , 0.0064 = 5^{-4} \times 4$

$0.4304_{10} = 0.2034_{5}$

Método de restas sucesivas para fracciones

- Ejercicios en clase:
  - $0.1250_{10} \rightarrow ?_{2}$
  - $0.4300_{10} \rightarrow ?_{8}$
  - $0.2500_{10} \rightarrow ?_{5}$
  - $0.6350_{10} \rightarrow ?_{16}$

Método del resto de cocientes para fracciones

Represar 0.4304_{10} en base 5

0.4304 x 5 = 2.1520
0.1520 x 5 = 0.7600
0.7600 x 5 = 3.8000
0.8000 x 5 = 4.0000

$0.4304_{10} = 0.2034_5$

Método del resto de cocientes para fracciones

- Ejercicios en clase:
  - $0.1250_{10} \rightarrow ?_{2}$
  - $0.4300_{10} \rightarrow ?_{8}$
  - $0.2500_{10} \rightarrow ?_{5}$
  - $0.6350_{10} \rightarrow ?_{16}$

Inexactitud para fracciones

Las fracciones no siempre pueden ser convertidas en forma exacta, con una expresión finita.

$$(0.3)_3 = 2 * 3^{-1} = 2 + 1/3$$

En definitiva, las fracciones en una base sólo pueden ser estimadas como una expresión finita en otra.

Conversión de potencias de 2

Para convertir números de base 2 a base k, donde k puede expresarse como una potencia de 2, es decir, $k=2^x$ donde $x>1$ y es un número entero, se llevan a cabo los siguientes pasos:

1. Se agrupan de x en x los dígitos que se encuentran a la izquierda del punto, comenzando a partir de él y aumentando ceros a la izquierda cuando es necesario.
2. Se agrupan de x en x los dígitos que se encuentran a la derecha del punto comenzando a partir de éste y aumentando ceros a la derecha cuando sea necesario.
3. Se sustituyen los grupos por los dígitos correspondientes en la base k.

Conversión de potencias de 2

Convertir $1110010100.011011_2 \rightarrow ?_{16}$

$16 = 2^4$

0011 1001 0100 . 0110 1100 Se agregaron dos ceros a la derecha

3 9 4 6 C

Por lo tanto: $1110010100.011011_2 \rightarrow 394.6C_{16}$

Conversión de potencias de 2

Para convertir números de base $k=2^x$ a base 2, se sustituye cada dígito en base k por los x dígitos binarios correspondientes.

$7402.61_8 \rightarrow ?_2$

$8 = 2^3$

7 4 0 2 . 6 1

111100 000 010 110 001

$7402.61_8 \rightarrow 111100000010.110001_2$

Conversiones ente sistemas o bases

Resumen de Conversión entre Sistemas de Numeración

	Binario	Octal	Decimal	Hexadecimal
Binario		Trios	Representación Polinomial	Cuartetos
Octal	Trios		Representación Polinomial	Octal->Dec.->Hexa Octal->Bin->Hexa
Decimal	Divisiónes Sucesivas(2)	Divisiónes Sucesivas(8)		Divisiónes Sucesivas(16)
Hexadecimal	Cuartetos	Hexa->Dec.->Octal Hexa->Bin.->Octal	Representación Polinomial	

Imagen tomada de: https://jjmejia06.wordpress.com/sistemas-de-numeracion-2/

Conversiones ente sistemas o bases

Base 10	Base 2	Base 8
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

Conversiones ente sistemas o bases

Base 10	Base 2	Base 16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0117	7

Base 10	Base 2	Base 16
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Operaciones aritméticas en base 2

Suma

+	0	1
0	0	1
1	1	10

Carry o acarreo

```
  111
  101101
+ 10110
--------
 1000011
```

Resta

-	0	1
0	0	1
1	1	0

Borrow (deber uno)

```
   101101
 - 10110
 -------
   010111
```

Multiplicación

*	0	1
0	0	0
1	0	1

```
101101*1101
  111
  101101
 1
 000000
1
 101101
1
 101101
---------
1001001001
```

División

/	0	1
0	e	0
1	e	1

```
      1000001
1101 1101010110
    -1101
     0000010110
       - 1101
         01001
```

Conversión de números con signo

- Para la **magnitud con signo**, el bit más a la izquierda es usado como indicador de signo. 1 implica que el número es negativo, 0 que es positivo.
  - Si se usan números de 8 bits, se va a poder representar el rango $[-(2^7-1), 2^7-1]$. En general, sobre números de n bits se puede representar el rango $[-(2^{(n-1)}-1), 2^{(n-1)}-1]$.
- La suma es igual que en el sistema decimal, incluyendo el concepto de acarreo.

Conversión de números con signo

			1	1	1	1		⇐ carries
0		1	0	0	1	1	1	(79)
0	+	0	1	0	0	0	1	1	+ (35)
0		1	1	1	0	0	1	0	(114)

En caso de llegar con un acarreo al bit 8, existe una situación de overflow.

Last carry	1	←		1	1	1	1		⇐ carries
overflows and is	0		1	0	0	1	1	1	1	(79)
discarded.	0	+	1	1	0	0	0	1	1	+ (99)
	0		0	1	1	0	0	1	0	(50)

Conversión de números con signo

- El complemento de un número se obtiene restándo dicho número al número más grande que puede representarse con el tamaño del número o palabra con que se cuenta.
- El acarreo se suma al dígito menos significativo.
- El complemento a r en d dígitos del número N es $(r^d-1)-N$.

Conversión de números con signo

$$-52_{10} = 999_{10} - 52_{10} = 947_{10}$$

$$167_{10} - 52_{10} = 167_{10} + 947_{10}$$

$$= 114_{10} + 1_{10} = 115_{10}$$

Conversión de números con signo

- Para el **complemento a 1**, la idea es igual a la vista en base 10. En este caso, el rango representado es el mismo que el que se tiene para una magnitud con signo.

$$-101_2 = 1111_2 - 0101_2 = 1010_2$$

Nota: el complemento a 1 de un número binario es invertir los dígitos.

Conversión de números con signo

- En el caso del **complemento a 2**, la idea detrás de este método es la misma que se presentó como **complemento a 1**, salvo que se realizerá la resta sobre el menor número que resulta mayor a todos los números representables con el tamaño de número o palabra con el que se cuenta:
  - El rango representable con complemento a 2 es $[-(2^{n-1}), 2^{n-1}-1]$
  - El complemento a r en d dígitos del número N es r^d-N si N distinto de 0 y 0 en caso contrario.

Conversión de números con signo

$$-101_2 = 10000_2 - 00101_2 = 1011_2$$

Nota: complemento a 2 no es más que **complemento a 1**, más 1.

Representación decimal de un número en complemento a 2

- Para los **positivos** es trivial, se hace lo que visto anteriormente, evaluando el polinomio que corresponda a la expresión binaria.

Algoritmo de suma utilizando la representación de números negativos mediante signo y magnitud

1.Sean $a_n a_{n-1} \ldots a_0$ y $b_n b_{n-1} \ldots b_0$ 2 números binarios con signo y magnitud.

2.Tienen signos iguales ? ($a_n = b_n$)

Si: sumar magnitudes quedando el resultado en $c_{n-1} c_{n-2} \ldots c_0$, $c_n \leftarrow b_n \leftarrow a_n$

No: Compar magnitudes y dejar en c_n el signo del mayor. Restar a la magnitud mayor la menor y el resultado queda en $c_{n-1} c_{n-2} \ldots c_0$

3. La magnitud de $c_{n-1} c_{n-2} \ldots c_0$ excede el rango ?

Si: Indicar error (overflow – sobreflujo)

No: El resultado está en $c_n c_{n-1} \ldots c_0$

Algoritmo de suma utilizando la representación de números negativos mediante signo y magnitud

Obtener el resultado de las siguientes sumas binarias a 4 dígitos:

- $5_{10}+(-3)_{10} = 0101_2+1011_2 = 0010$, su equivalente decimal es 2
  - Los signos son diferentes, y la magnitud del primer número es > que la del segundo, así que restamos 011 de 101 y el signo del resultado será positivo

- $-4_{10} + (-6)_{10} = 1100_2 + 1110_2 = 1010_2$

  +100
  1110
  1010

  Overflow

  - Los signos son iguales, así que se suman magnitudes

    Error ! Existe overflow

Nota: El Overflow se genera cuando ya no hay lugar para un dígito más. En base binaria corresponde a un cambio de signo

Algoritmo de suma algebraica en complemento a 1

1. Tomar el complemento a 1 de los números negativos
2. Sumar los operandos
3. Existe carry? Si: sumar 1 al resultado
4. Existe overflow? Si: indicar error

   No: Escribir el resultado

Utilizando 4 dígitos

$(-4)_{10} + (-3)_{10} = 0100_{2^{c1}} + 0011_{2^{c1}} = 1011_2 + 1100_2 = 1\ 0111$ Existe carry

$= 0111_2 + 1_2 = 1000_2$ No existe overflow

Algoritmo de suma algebraica en complemento a 2

1. Tomar el complemento a 2 de los números negativos
2. Sumar los operandos
3. Existe overflow? Si: mensaje de error

   No: Se toman las primeras n posiciones de derecha a izquierda como resultado ignorando el carry si es que lo hay.

Tomando 4 dígitos

$7_{10} + (-5)_{10} = 0111_2 + 0101_2{}^{c2} = 0111_2 + 1011_2 = 1\ 0010_2$

Existe carry, así que el resultado es 0010

Representación decimal de un número en complemento a 2

- En el caso de los **negativos**, se debe hacer el procedimiento inverso
  - Se invierten los bits,
  - Se suma 1,
  - Se convierte a decimal
  - Se le coloca el signo – adelante
- Esto es equivalente a obtener el complemento a 1 y sumarle 1.

Actividades reto

- Realice en lenguaje C un programa que permita hacer conversiones de números entre las distintas bases.
- Realice otro programa en C para realizar sumas, restas, multiplicaciones y divisiones de valores binarios.

Para obtener los programas, favor de escribir a: hrorozcoa@uaemex.mx

Referencias

- Null, L. and J. Lobur. The Essentials of Computer Organization and Architecture, Jones and Bartlett Publishers, Feb. 2003
- Kip, I. Lenguaje ensamblador para computadoras basadas en Intel, Pearson, Quinta edición.
- Willian H. Murray III Chris H. Pappas. Programación en lenguaje ensamblador, McGraw-Hill.