UNIVERSIDAD NACIONAL DE CÓRDOBA

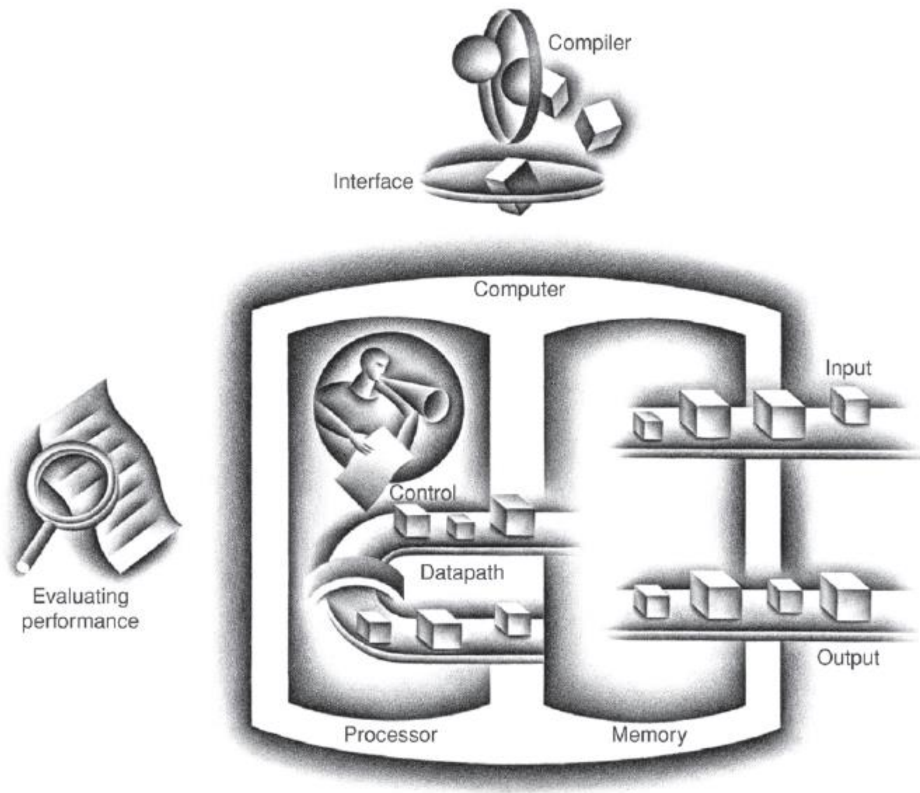
FACULTAD DE MATEMÁTICA ASTRONOMÍA, FÍSICA Y COMPUTACIÓN.

ORGANIZACIÓN DEL COMPUTADOR

TEÓRICOS: PABLO A. FERREYRA

PRÁCTICOS:
DELFINA VELEZ
AGUSTÍN LAPROVITA
GONZALO VODANOVICK

CÁTEDRA DE ORGANIZACIÓN DEL COMPUTADOR



- **Aprenderemos las partes constitutivas de las computadoras, y su hardware y software, partiende desde los sistemas combinacionales y los secuenciales.**

ACTIVIDADES Y CRONOGRAMA

- I . DOS PARCIALES Y DOS RECUPERATORIOS
- II. LABORATORIOS
- III . PROYECTOS FINALES

EL CRONOGRAMA CON LA DESCRIPCIÓN DE TAREAS SE SUBIRÁ A MOODLE, SEMANALMENTE.

SE PROMOCIONA CON DOS PARCIALES APROBADOS, PROMEDIO SUPERIOR A SIETE NOTA MAYOR A SEIS. LOS LABORATORIOS APROBADOS Y LOS PROYECTOS FINALES APROBADOS.

PARA REGULARIDAD DOS PARCIALES APROBADOS. MÁS LABORATORIOS Y PROYECTOS EN VERSIONES REDUCIDAS.

LOS PARCIALES SE APRUEBAN CON 4 (CUATRO). LOS PROYECTOS PODRÍAN EVENTUALMENTE SUMAR NOTA A LOS PARCIALES, (0,1,2).

PROGRAMA

UNC Universidad Nacional de Córdoba

FAMAF Facultad de Matemática, Astronomía, Física y Computación

EXP-UNC 23224/2016 Res. CD N° 141/2016

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Organización del Computador	**AÑO:** 2016
CARACTER: Obligatoria	**UBICACIÓN EN LA CARRERA:** 2° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
REGIMEN: Cuatrimestral	**CARGA HORARIA:** 120 horas

FUNDAMENTACIÓN Y OBJETIVOS

Que el alumno sea capaz de reconocer las unidades constitutivas de un sistema de computación y comprender su funcionamiento interno, como así también la interacción entre ellas. Que se inicie en la programación en Assembly.

PROGRAMA

Aritmética Binaria

- Sistemas binarios de numeración.
- Representación de números negativos.
- Puntos fijo y flotante.
- Máquinas algorítmicas para aritmética binaria.
- Errores en la representación de los datos a nivel máquina.

PROGRAMA

CONTENIDO

Circuitos Lógicos Combinacionales

- Funciones lógicas. Postulados y propiedades del álgebra de conmutación (Boole).
- Minimización mediante el uso de los mismos.
- Circuitos lógicos de bajo y medio nivel de integración.

Circuitos Lógicos Secuenciales

- Celda básica de memoria ("Flip-Flop D").
- Circuitos lógicos secuenciales sincrónicos.
- Autómatas de Mealy y Moore.
- Introducción a los circuitos lógicos secuenciales programables.
- "Latchs" y "Shift Registers".

PROGRAMA

Sistemas de Memoria

- Conceptos fundamentales sobre memorias "Read Only Memory" - ROM, "Programmable Read Only Memory" - PROM, "Erasable Programmable Only Memory" - EPROM y "Electricaly Erasable Programmable Read Only Memory" - EEPROM (Introducción a los "Programmable Logic Devices" - PLD). Memoria "FLASH".
- Conceptos fundamentales sobre memorias "Random Access Memory" - RAM estáticas (SRAM) y dinámicas (DRAM).
- Estructuración o decodificado de bancos de memorias ("Memory Mapped").
- Otros tipos de Memorias. Ancho de banda.

PROGRAMA

Procesadores tipo Von Newman y Harvard

- Líneas de direccionamiento, datos y control.
- Registros internos.
- Modos de direccionamientos.
- Instrucciones (Incluye conceptos sobre lenguaje "assembly").
- Interrupciones y Excepciones.

PROGRAMA

Sistemas de Entradas Salida

"2016 - Año del Bicentenario de la Declaración de la Independencia Nacional"

UNC Universidad Nacional de Córdoba

FAMAF Facultad de Matemática, Astronomía, Física y Computación

EXP-UNC 23224/2016 Res. CD N° 141/2016

- Nociones de Puertos Paralelos, su estructura y utilización.
- Nociones de Puertos Seriales, su estructura y utilización.

PROGRAMA

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

1.-David A. Patterson and John L Hennessy: "Computer Organization and Design – The Hardware/Software Interface". Fourth Edition. Elsevier – Morgan Kaufmann (ISBN 978-012-374493-7)

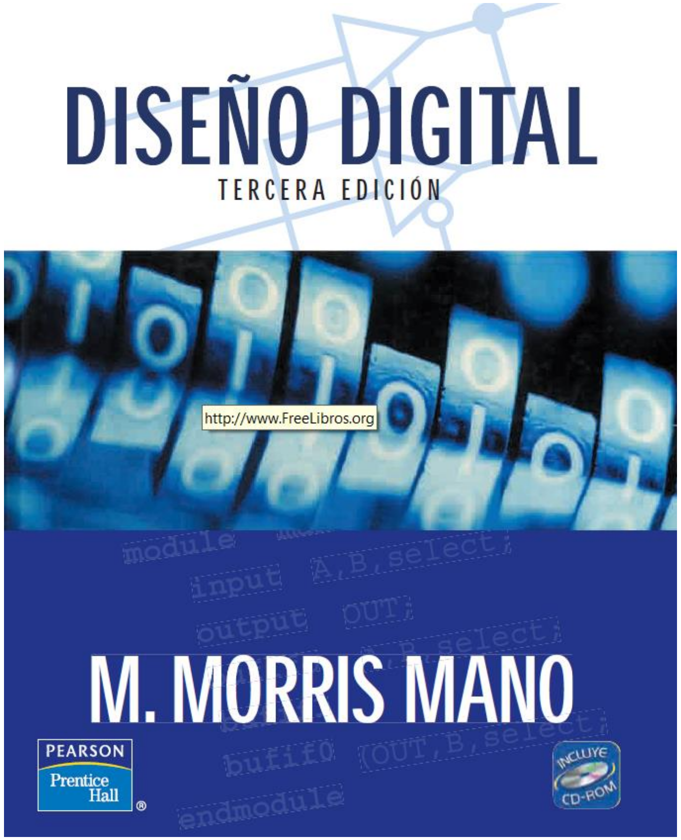
2.-Morris Mano, M.: "Ingeniería Computacional, diseño del hardware". Prentice Hall Hispanoamericana S.A., 1992.

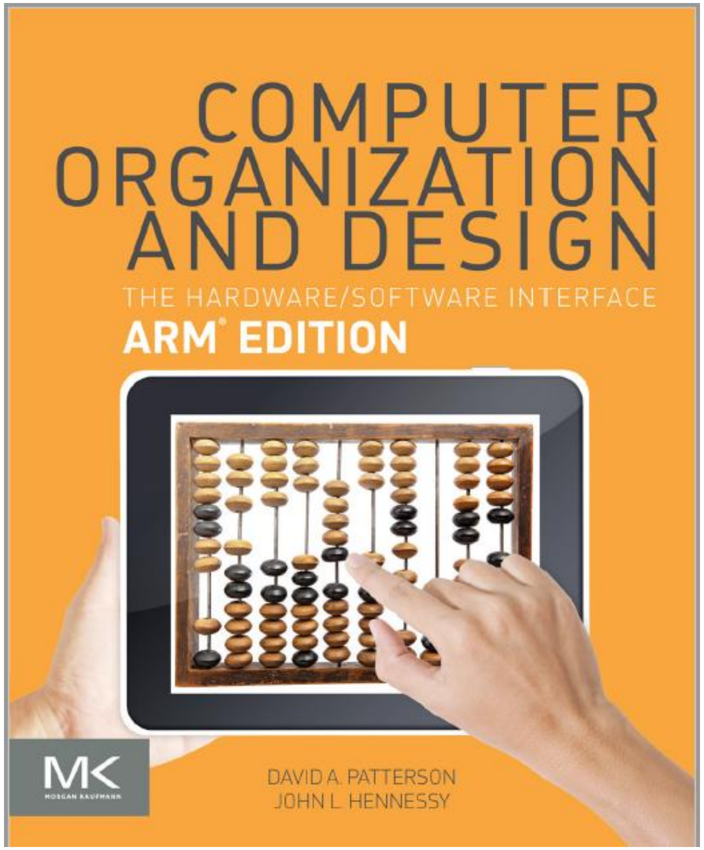
BIBLIOGRAFÍA COMPLEMENTARIA

3.-Tanenbaum, A. S.: "Organización de Computadoras, un enfoque estructurado". Prentice Hall Hispanoamericana S. A., 1992.

4-Thomas C. Bartee: "Fundamentos de Computadoras Digitales". Mc. Graw Hill, quinta edición (Primera en castellano).

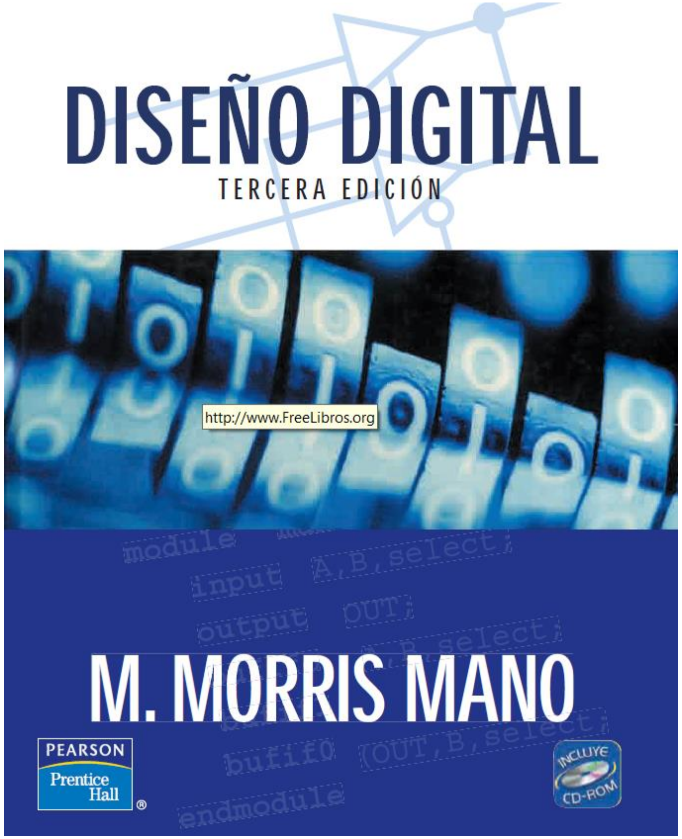
BIBLIOGRAFÍA





- SISTEMAS DE NUMERACIÓN

T1: SISTEMAS DE NUMERACIÓN



T1: SISTEMAS DE NUMERACIÓN

DISEÑO DIGITAL

TERCERA EDICIÓN

M. Morris Mano
CALIFORNIA STATE UNIVERSITY, LOS ANGELES

TRADUCCIÓN
Roberto Escalona García
Ingeniero Químico
Universidad Nacional Autónoma de México

REVISIÓN TÉCNICA
Gonzalo Duchén Sánchez
Sección de Estudios de Postgrado e Investigación
Escuela Superior de Ingeniería Mecánica y Eléctrica
Unidad Culhuacán
Instituto Politécnico Nacional

PEARSON Educación®

México • Argentina • Brasil • Colombia • Costa Rica • Chile • Ecuador
España • Guatemala • Panamá • Perú • Puerto Rico • Uruguay • Venezuela

T1: SISTEMAS DE NUMERACIÓN

CONTENIDO

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

Un número decimal, como 7,392, representa una cantidad igual a 7 millares más 3 centenas, más 9 decenas, más 2 unidades. Los millares, centenas, etcétera, son potencias de 10 que están implícitas en la posición de los coeficientes. Si queremos ser más exactos, deberíamos escribir 7,392 así:

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

No obstante, por convención, se escriben únicamente los coeficientes y se deducen las potencias necesarias de 10 de la posición que dichos coeficientes ocupan. En general, un número con punto decimal se representa con una serie de coeficientes, así:

$$a_5 a_4 a_3 a_2 a_1 a_0 \cdot a_{-1} a_{-2} a_{-3}$$

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

Los coeficientes a_j son cualesquiera de los 10 dígitos (0, 1, 2, . . . , 9); el valor del subíndice j indica el valor de posición y, por tanto, la potencia de 10 por la que se deberá multiplicar ese coeficiente. Esto puede expresarse así:

$$10^5 a_5 + 10^4 a_4 + 10^3 a_3 + 10^2 a_2 + 10^1 a_1 + 10^0 a_0 + 10^{-1} a_{-1} + 10^{-2} a_{-2} + 10^{-3} a_{-3}$$

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

Decimos que el sistema numérico decimal es *base* 10 porque usa 10 dígitos y los coeficientes se multiplican por potencias de 10. El sistema *binario* es un sistema numérico diferente. Sus coeficientes sólo pueden tener dos valores: 0 o 1. Cada coeficiente a_j se multiplica por 2^j. Por ejemplo, el equivalente decimal del número binario 11010.11 es 26.75, como puede verse si multiplicamos los coeficientes por potencias de 2:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 26.75$$

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

En general, un número expresado en un sistema base r consiste en coeficientes que se multiplican por potencias de r:

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \ldots + a_2 \cdot r^2 + a_1 \cdot r + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \ldots + a_{-m} \cdot r^{-m}$$

El valor de los coeficientes a_j varía entre 0 y $r - 1$. Para distinguir entre números con diferente base, encerramos los coeficientes en paréntesis y añadimos un subíndice que indica la base empleada (aunque a veces se hace una excepción en el caso de los números decimales, si por el contexto es obvio que la base es 10). Un ejemplo de número base 5 es

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$$

Los valores de los coeficientes en base 5 sólo pueden ser 0, 1, 2, 3 y 4. El sistema numérico octal es un sistema base 8 que tiene ocho dígitos: 0, 1, 2, 3, 4, 5, 6 y 7. Un ejemplo de número octal es 127.4. Para determinar su valor decimal equivalente, expandimos el número como una serie de potencias con base 8:

$$(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$$

Advierta que los dígitos 8 y 9 no pueden aparecer en un número octal.

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

Se acostumbra tomar del sistema decimal los *r* dígitos requeridos si la base del número es menor que 10, y utilizar las letras del alfabeto para complementar los 10 dígitos decimales si la base del número es mayor que 10. Por ejemplo, en el sistema numérico *hexadecimal* (base 16), los primeros 10 dígitos se toman del sistema decimal, y se usan las letras A, B, C, D, E y F para los dígitos 10, 11, 12, 13, 14 y 15, respectivamente. He aquí un ejemplo de número hexadecimal:

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46{,}687)_{10}$$

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

Como ya se señaló, los dígitos de los números binarios se llaman *bits*. Si un bit es igual a 0, no contribuye a la suma durante la conversión. Por tanto, la conversión de binario a decimal puede efectuarse sumando los números con potencias de 2 correspondientes a los bits que son 1. Por ejemplo,

$$(110101)_2 = 32 + 16 + 4 + 1 = (53)_{10}$$

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

Tabla 1-1
Potencias de dos

n	2n	n	2n	n	2n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

T1: SISTEMAS DE NUMERACIÓN

1-2 NÚMEROS BINARIOS

sumando:	101101	minuendo:	101101	multiplicando:	1011
sumando:	+100111	sustraendo:	−100111	multiplicador:	× 101
suma:	1010100	diferencia:	000110		1011
					0000
					1011
				producto:	110111

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

	Cociente entero		Residuo	Coeficiente
$41/2 =$	20	+	$\frac{1}{2}$	$a_0 = 1$
$20/2 =$	10	+	0	$a_1 = 0$
$10/2 =$	5	+	0	$a_2 = 0$
$5/2 =$	2	+	$\frac{1}{2}$	$a_3 = 1$
$2/2 =$	1	+	0	$a_4 = 0$
$1/2 =$	0	+	$\frac{1}{2}$	$a_5 = 1$

Por tanto, la respuesta es $(41)_{10} = (a_5a_4a_3a_2a_1a_0)_2 = (101001)_2$

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

EJEMPLO 1-1

El proceso aritmético se puede plantear de forma más conveniente como sigue:

Entero	**Residuo**	
41		
20	1	
10	0	
5	0	
2	1	
1	0	
0	1	101001 = respuesta

La conversión de enteros decimales a cualquier sistema base *r* es similar a este ejemplo, sólo que se divide entre *r* en vez de entre 2.

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

EJEMPLO 1-2

Convertir 153 decimal a octal. La base *r* en este caso es 8. Primero dividimos 153 entre 8 para obtener un cociente entero de 19 y un residuo de 1. Luego dividimos 19 entre 8 para obtener un cociente entero de 2 y un residuo de 3. Por último, dividimos 2 entre 8 para obtener un cociente de 0 y un residuo de 2. Este proceso se puede plantear así:

$$\begin{array}{r|c} 153 & \\ 19 & 1 \\ 2 & 3 \\ 0 & 2 \end{array} \quad = (231)_8$$

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

La conversión de una *fracción* decimal a binario se efectúa con un método similar al que se utiliza con enteros, pero se multiplica en lugar de dividir y se acumulan enteros en vez de residuos. En este caso, también, la mejor explicación es un ejemplo.

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

EJEMPLO 1-3

Convertir $(0.6875)_{10}$ a binario. Primero, multiplicamos 0.6875 por 2 para obtener un entero y una fracción. La nueva fracción se multiplica por 2 para dar un nuevo entero y una nueva fracción. El proceso se continúa hasta que la fracción es 0 o hasta que se tienen suficientes dígitos para la precisión deseada. Los coeficientes del número binario se obtienen de los enteros, así:

	Entero		Fracción	Coeficiente
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	0.0000	$a_{-4} = 1$

Por tanto, la respuesta es $(0.6875)_{10} = (0.a_{-1}a_{-2}a_{-3}a_{-4})_2 = (0.1011)_2$

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

Para convertir una fracción decimal a un número expresado en base r, seguimos un procedimiento similar, multiplicando por r en vez de por 2. Los coeficientes obtenidos a partir de los enteros tendrán valores entre 0 y $r - 1$, en vez de ser sólo 0 y 1.

T1: SISTEMAS DE NUMERACIÓN

1-3 CONVERSIONES DE BASE NUMÉRICA

EJEMPLO 1-4

Convertir $(0.513)_{10}$ a octal.

$$0.513 \times 8 = 4.104$$
$$0.104 \times 8 = 0.832$$
$$0.832 \times 8 = 6.656$$
$$0.656 \times 8 = 5.248$$
$$0.248 \times 8 = 1.984$$
$$0.984 \times 8 = 7.872$$

La respuesta, con siete cifras significativas, se obtiene de la parte entera de los productos

$$(0.513)_{10} = (0.406517\ldots)_8$$

1-3 CONVERSIONES DE BASE NUMÉRICA

La conversión de números decimales que tienen tanto parte entera como parte fraccionaria se efectúa convirtiendo por separado las dos partes y combinando después las dos respuestas. Si usamos los resultados de los ejemplos 1-1 y 1-3, obtendremos

$$(41.6875)_{10} = (101001.1011)_2$$

De los ejemplos 1-2 y 1-4 tenemos

$$(153.513)_{10} = (231.406517)_8$$

T1: SISTEMAS DE NUMERACIÓN

1-4 NÚMEROS OCTALES Y HEXADECIMALES

Las conversiones entre binario, octal y hexadecimal desempeñan un papel importante en las computadoras digitales. Puesto que $2^3 = 8$ y $2^4 = 16$, cada dígito octal corresponde a tres dígitos binarios y cada dígito hexadecimal corresponde a cuatro dígitos binarios. En la tabla 1-2 se presentan los primeros 16 números de los sistemas numéricos decimal, binario, octal y hexadecimal.

T1: SISTEMAS DE NUMERACIÓN

1-4 NÚMEROS OCTALES Y HEXADECIMALES

Tabla 1-2
Números con diferente base

Decimal (base 10)	Binario (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

T1: SISTEMAS DE NUMERACIÓN

1-4 NÚMEROS OCTALES Y HEXADECIMALES

La conversión de binario a octal se efectúa fácilmente acomodando los dígitos del número binario en grupos de tres, partiendo del punto binario tanto a la izquierda como a la derecha. Luego, se asigna el dígito octal correspondiente a cada grupo. Este ejemplo ilustra el procedimiento:

$$\begin{array}{cccccccccc} (10 & 110 & 001 & 101 & 011 & \cdot & 111 & 100 & 000 & 110)_2 = (26153.7460)_8 \\ 2 & 6 & 1 & 5 & 3 & & 7 & 4 & 0 & 6 \end{array}$$

T1: SISTEMAS DE NUMERACIÓN

1-4 NÚMEROS OCTALES Y HEXADECIMALES

La conversión de binario a hexadecimal es similar, sólo que el número binario se divide en grupos de cuatro dígitos:

$$(10 \quad 1100 \quad 0110 \quad 1011 \quad \cdot \quad 1111 \quad 0010)_2 = (2C6B.F2)_{16}$$

$$\begin{array}{cccccc} 2 & C & 6 & B & F & 2 \end{array}$$

T1: SISTEMAS DE NUMERACIÓN

1-4 NÚMEROS OCTALES Y HEXADECIMALES

La conversión de octal o hexadecimal a binario se hace invirtiendo el procedimiento anterior. Cada dígito octal se convierte a su equivalente binario de tres dígitos. Asimismo, cada dígito hexadecimal se convierte en su equivalente binario de cuatro dígitos. Los ejemplos siguientes ilustran el procededimiento:

$$(673.124)_8 = (\underset{6}{110} \quad \underset{7}{111} \quad \underset{3}{011} \cdot \underset{1}{001} \quad \underset{2}{010} \quad \underset{4}{100})_2$$

$$(306.D)_{16} = (\underset{3}{0011} \quad \underset{0}{0000} \quad \underset{6}{0110} \cdot \underset{D}{1101})_2$$

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base disminuida

Dado un número N en base r que tiene n dígitos, el complemento a $(r - 1)$ de N se define como $(r^n - 1) - N$. En el caso de números decimales, $r = 10$ y $r - 1 = 9$, así que el complemento a nueve de N es $(10^n - 1) - N$. En este caso, 10^n representa un número que consiste en un uno seguido de n ceros. $10^n - 1$ es un número representado por n nueves. Por ejemplo, si $n = 4$, tenemos $10^4 = 10{,}000$ y $10^4 - 1 = 9999$. De esto se sigue que el complemento a nueve de un número decimal se obtiene restando cada dígito a nueve. He aquí algunos ejemplos numéricos:

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base disminuida

El complemento a nueve de 546700 es $999999 - 546700 = 453299$.

El complemento a nueve de 012398 es $999999 - 012398 = 987601$.

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base disminuida

En el caso de los números binarios, $r = 2$ y $r - 1 = 1$, así que el complemento a uno de N es $(2^n - 1) - N$. Aquí también, 2^n se representa con un número binario que consiste en un uno seguido de n ceros. $2^n - 1$ es un número binario representado por n unos. Por ejemplo, si $n = 4$, tenemos $2^4 = (10000)_2$ y $2^4 - 1 = (1111)_2$. Así, el complemento a uno de un número binario se obtiene restando cada dígito a uno. Sin embargo, al restar dígitos binarios a 1 podemos tener $1 - 0 = 1$ o bien $1 - 1 = 0$, lo que hace que el bit cambie de 0 a 1 o de 1 a 0. Por tanto, el complemento a uno de un número binario se forma cambiando los unos a ceros y los ceros a unos. He aquí algunos ejemplos numéricos:

1-5 COMPLEMENTOS

Complemento a la base disminuida

El complemento a uno de 1011000 es 0100111.

El complemento a uno de 0101101 es 1010010.

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base

El complemento a r de un número N de n dígitos en base r se define como $r^n - N$, para $N \neq 0$, y 0 para $N = 0$. Si comparamos con el complemento a $(r - 1)$, veremos que el complemento a r se obtiene sumando 1 al complemento a $(r - 1)$, ya que $r^n - N = [(r^n - 1) - N] + 1$. Así pues, el complemento a 10 del número decimal 2389 es $7610 + 1 = 7611$, y se obtiene sumando 1 al valor del complemento a nueve. El complemento a dos del número binario 101100 es $010011 + 1 = 010100$, y se obtiene sumando 1 al valor del complemento a uno.

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base

Puesto que 10^n es un número que se representa con un uno seguido de n ceros, $10^n - N$, que es el complemento a 10 de N, también puede formarse dejando como están todos los ceros menos significativos, restando a 10 el primer dígito menos significativo distinto de cero, y restando a 9 los demás dígitos a la izquierda.

El complemento a 10 de 012398 es 987602.

El complemento a 10 de 246700 es 753300.

El complemento a 10 del primer número se obtiene restando 8 a 10 en la posición menos significativa y restando a 9 todos los demás dígitos. El complemento a 10 del segundo número se obtiene dejando como están los dos ceros de la derecha, restando 7 a 10 y restando a 9 los otros tres dígitos.

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base

De forma similar, el complemento a dos se forma dejando como están todos los ceros menos significativos y el primer uno, y sustituyendo los unos por ceros y los ceros por unos en las demás posiciones a la izquierda.

El complemento a dos de 1101100 es 0010100.

El complemento a dos de 0110111 es 1001001.

T1: SISTEMAS DE NUMERACIÓN

1-5 COMPLEMENTOS

Complemento a la base

PUNTO FIJO

En las definiciones anteriores se supuso que los números no llevan punto. Si el número N original lleva punto, deberá quitarse temporalmente para formar el complemento a r o a $(r - 1)$, y volver a colocarlo después en el número complementado en la misma posición relativa. También vale la pena mencionar que el complemento del complemento restablece el valor original del número. El complemento a r de N es $r^n - N$. El complemento del complemento es $r^n - (r^n - N) = N$, o sea, el número original.

Floating Point

- Representation for non-integral numbers
  - Including very small and very large numbers
- Like scientific notation
  - -2.34×10^{56}
  - $+0.002 \times 10^{-4}$
  - $+987.02 \times 10^{9}$

normalized

not normalized

- In binary
  - $\pm 1.xxxxxxx_2 \times 2^{yyyy}$
- Types `float` and `double` in C

Floating Point Standard

- Defined by IEEE Std 754-1985
- Developed in response to divergence of representations
  - Portability issues for scientific code
- Now almost universally adopted
- Two representations
  - Single precision (32-bit)
  - Double precision (64-bit)

IEEE Floating-Point Format

	single: 8 bits double: 11 bits	single: 23 bits double: 52 bits
S	Exponent	Fraction

$$x = (-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$$

- S: sign bit (0 ⇒ non-negative, 1 ⇒ negative)
- Normalize significand: $1.0 \leq |\text{significand}| < 2.0$
  - Always has a leading pre-binary-point 1 bit, so no need to represent it explicitly (hidden bit)
  - Significand is Fraction with the "1." restored
- Exponent: excess representation: actual exponent + Bias
  - Ensures exponent is unsigned
  - Single: Bias = 127; Double: Bias = 1203

Single-Precision Range

- Exponents 00000000 and 11111111 reserved
- Smallest value
  - Exponent: 00000001
    $\Rightarrow$ actual exponent = 1 – 127 = –126
  - Fraction: 000…00 $\Rightarrow$ significand = 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$
- Largest value
  - exponent: 11111110
    $\Rightarrow$ actual exponent = 254 – 127 = +127
  - Fraction: 111…11 $\Rightarrow$ significand $\approx$ 2.0
  - $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

Double-Precision Range

- Exponents 0000…00 and 1111…11 reserved
- Smallest value
  - Exponent: 00000000001
    $\Rightarrow$ actual exponent = 1 – 1023 = –1022
  - Fraction: 000…00 $\Rightarrow$ significand = 1.0
  - $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$
- Largest value
  - Exponent: 11111111110
    $\Rightarrow$ actual exponent = 2046 – 1023 = +1023
  - Fraction: 111…11 $\Rightarrow$ significand $\approx$ 2.0
  - $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$

Floating-Point Precision

- Relative precision
  - all fraction bits are significant
  - Single: approx 2^{-23}
    - Equivalent to $23 \times \log_{10}2 \approx 23 \times 0.3 \approx 6$ decimal digits of precision
  - Double: approx 2^{-52}
    - Equivalent to $52 \times \log_{10}2 \approx 52 \times 0.3 \approx 16$ decimal digits of precision

Floating-Point Example

- Represent –0.75
  - $-0.75 = (-1)^1 \times 1.1_2 \times 2^{-1}$
  - S = 1
  - Fraction = $1000\ldots00_2$
  - Exponent = –1 + Bias
    - Single: $-1 + 127 = 126 = 01111110_2$
    - Double: $-1 + 1023 = 1022 = 01111111110_2$
- Single: 1011111101000…00
- Double: 1011111111101000…00

Floating-Point Example

- What number is represented by the single-precision float
  11000000101000…00
  - S = 1
  - Fraction = $01000\ldots00_2$
  - Fxponent = $10000001_2 = 129$
- $x = (-1)^1 \times (1 + 01_2) \times 2^{(129 - 127)}$
  $= (-1) \times 1.25 \times 2^2$
  $= -5.0$

Infinities and NaNs

- Exponent = 111...1, Fraction = 000...0
  - ±Infinity
  - Can be used in subsequent calculations, avoiding need for overflow check
- Exponent = 111...1, Fraction ≠ 000...0
  - Not-a-Number (NaN)
  - Indicates illegal or undefined result
    - e.g., 0.0 / 0.0
  - Can be used in subsequent calculations

Floating-Point Addition

- Consider a 4-digit decimal example
  - $9.999 \times 10^1 + 1.610 \times 10^{-1}$
- 1. Align decimal points
  - Shift number with smaller exponent
  - $9.999 \times 10^1 + 0.016 \times 10^1$
- 2. Add significands
  - $9.999 \times 10^1 + 0.016 \times 10^1 = 10.015 \times 10^1$
- 3. Normalize result & check for over/underflow
  - 1.0015×10^2
- 4. Round and renormalize if necessary
  - 1.002×10^2

Floating-Point Addition

- Now consider a 4-digit binary example
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + −0.4375)
- 1. Align binary points
  - Shift number with smaller exponent
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
- 2. Add significands
  - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
- 3. Normalize result & check for over/underflow
  - $1.000_2 \times 2^{-4}$, with no over/underflow
- 4. Round and renormalize if necessary
  - $1.000_2 \times 2^{-4}$ (no change) $= 0.0625$