Introducción al Diseño Lógico (E0301)

Ingeniería en Computación

Gerardo Sager

Clase 2 curso 2024

Parte I

- Conversión entre sistemas numéricos
  - Decimal, binario, hexadecimal.
- Ventajas del uso de hexadecimal
  - Conteo en hexadecimal.
- Representación de números decimales usando código BCD.
- Códigos Alfanuméricos: código ASCII.
- Paridad y su utilización para la detección de errores.

Conversión Binaria a Decimal

- Convertir de binario a decimal sumando las posiciones que contienen 1 con su peso correspondiente:

$$\mathbf{11011_2}$$

$$11011_2 = 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 =$$

$$= 16+8+2+1 = 27$$

- Un ejemplo con más bits

$$\mathbf{10110101_2}$$

- $10110101_2 = 1x2^7 + 1x2^5 + 1x2^4 + 1x2^2 + 1x2^0 =$

$$= 128+32+16+4+1=181$$

Conversión Binaria a Decimal

- Los números binarios pueden convertirse con el método “double-dabble”

$$
\begin{array}{llllll}
\text{Given:} & 1 & 1 & 0 & 1 & 1_2 \\
\text{Results:} & 1 \times 2 = 2 & & & & \\
 & \underline{+\,1} & & & & \\
 & & 3 \times 2 = 6 & & & \\
 & & \underline{+\,0} & & & \\
 & & & 6 \times 2 = 12 & & \\
 & & & \underline{+1} & & \\
 & & & & 13 \times 2 = 26 & \\
 & & & & \underline{+1} & \\
 & & & & & 27_{10}
\end{array}
$$

El método double-dabble permite la conversión sin hacer la suma de grandes números

Conversión Binaria a Decimal

Los números binarios pueden convertirse a decimal con el método "double-dabble"

Dado:	**1**		**1**		**0**		**1**		**1**
Resultado	1	x2 =	2						
		+	1						
			3	x2 =	6				
				+	0				
					6	x2 =	12		
						+	1		
							13	x2 =	26
								+	1
									27_{10}

El método double-dabble permite la conversión sin hacer la suma de grandes números

Conversión Binaria a Decimal

- Proceso inverso a la suma de potencias pesadas.
- Debo tener una tabla de potencias de 2 expresadas en decimal.
- Me fijo cual es el mayor número de la tabla, contenido en el número a convertir.
- Resto del número a convertir el valor obtenido
- Repito el proceso hasta llegar a 0
- Establezco en 1 el bit correspondiente si usé el valor de la tabla y en 0 si no lo usé

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

Ejemplo: 314_{10} = ? → 314-256=58 (2^8) → 58-32=26 (25) → 26-16=10 (2^4)
→ 10-8=2 (23) → 2-2=0 (2^1) → $314_{10}=100111010_2$

División Repetida

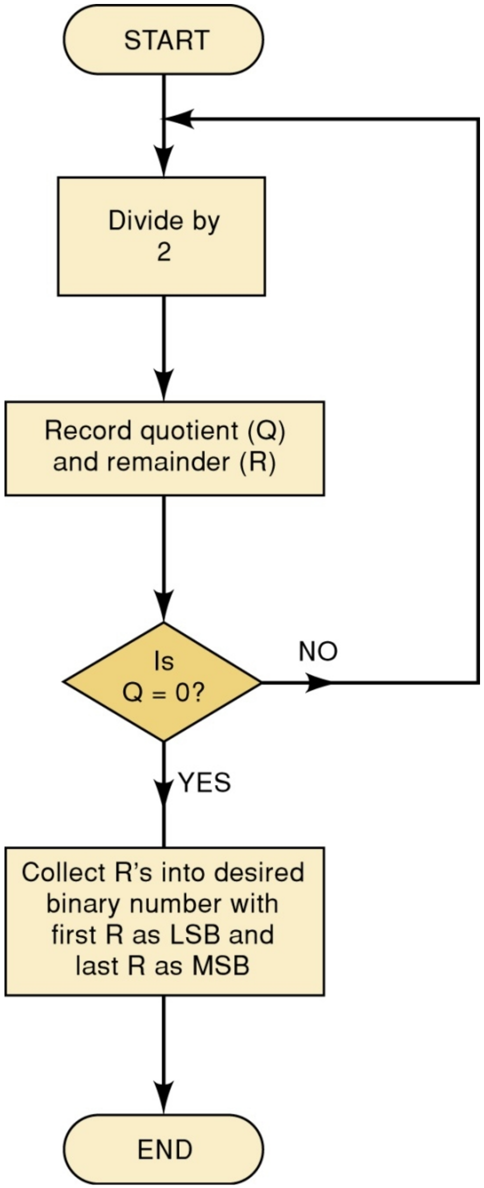
- Dividir el número decimal por 2.
- Escribir el resto de cada división y volver a dividir el cociente x 2 hasta que se obtenga un cociente = 0
- El primer resto es el LSB
- El último resto es el MSB

314/2	Q=157	R=0 (LSB)
157/2	Q=78	R=1
78/2	Q=39	R=0
39/2	Q=19	R=1
19/2	Q=9	R=1
9/2	Q=4	R=1
4/2	Q=2	R=0
2/2	Q=1	R=0
1/2	Q=0	R=1 (MSB)

$314_{10} = 100111010_2$

División Repetida

- Este diagrama de flujo describe el proceso a realizar para convertir de Decimal a Binario.
- Si en vez de dividir por 2 se divide por N, me permite convertir de decimal a base N



Sistema de Numeración Hexadecimal

- Hexadecimal permite un manejo conveniente de cadenas binarias largas, ya que 4 bits agrupados pueden representarse como un único dígito hexadecimal.(base 16)
- Los símbolos que representan los dígitos hexadecimales, son 0-9 y A-F
- La representación puede ser entera o fraccionaria.

16^4	16^3	16^2	16^1	16^0	16^{-1}	16^{-2}	16^{-3}	16^{-4}

Hexadecimal point

Relación entre números hexadecimales, decimales, y binarios.

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Sistema de Numeración Hexadecimal– Hexa a Decimal

- Convertir de hexa a decimal multiplicando cada dígito por su peso posicional.

- En el siguiente ejemplo, E se sustituye por 14 y A por 10 para realizar la suma pesada.

$$E2A_{16} = E \times 256 + 2 \times 16 + 10 = ...$$

Para practicar, verificar que $3BC2_{16}$ is igual a ….

Sistema de Numeración Hexadecimal – Decimal a Hexa

- Se pueden usar varios métodos: convertir a binario y luego agrupar en dígitos hexadecimales.
- Usar el método de división repetida
  - Dividir el número decimal por 16
  - El primer resto es el LSB
  - El último es el MSB.
- Ejemplo:
- 100000_{10}

Sistema de Numeración Hexadecimal – Hexa a Binario

- Se convierte cada dígito Hexa directamente a binario y se reescribe.
- Si es necesario se pueden agregar ceros a la izquierda para completar una palabra
- $19F2_{16}$ =0001 1001 1111 0010

Para practicar verificar que $BA6_{16} = 101110100110_2$

- Convertir de binario a Hexa, agrupando bits de a cuatro comenzando por el LSB.
  - Luego cada grupo se escribe como un único digito hexa equivalente

$$1\,1\,1\,0\,1\,0\,0\,1\,1\,0_2 = \underbrace{\boxed{0\ 0}\ 1\ 1}_{3}\ \underbrace{1\ 0\ 1\ 0}_{A}\ \underbrace{0\ 1\ 1\ 0}_{6}$$

$$= 3A6_{16}$$

Verificar que 101011111 $_2$ = …

Codificación BCD

- La codificación BCD se usa ampliamente, sobre todo en equipos de bajo costo, para representar números decimales en forma binaria
  - Combina características de los sistemas decimal y binario.
    - Cada dígito decimal se convierte a su equivalente binario
- BCD **NO ES** un sistema numérico.
  - Es un número decimal, con cada dígito codificado como su equivalente en binario.
- Un número BCD no es lo mismo que un número binario.
  - La ventaja principal de BCD es que resulta fácil convertir de BCD a decimal y viceversa.

Codificación BCD

- Convertir el número 874_{10} a BCD:
  - Cada dígito se representa mediante 4 bits.
  - El ´numero representado mediante esos 4 bits no puede ser superior a 9

1	0	0	0	0	1	1	1	0	1	0	0

8 **7** **4**

- Para convertir de BCD a decimal se realiza el proceso inverso:
  - Se agrupan los bits de a 4.
  - Se determina el valor decimal de cada grupo de 4 bits, que deben estar comprendidos entre 0 y 9

- Comparación entre BCD y Binario.
  - Cual es el rango de valores que pueden representarse en BCD con 16 bits?
  - Cual es el rango de valores que pueden representarse en binario con 16 bits?
- Convertir 0111010000111001 a su equivalente decimal considerando primero que es BCD y luego que es binario.
  - Cuál de las dos conversiones es mas sencilla?

Byte, Nibble, Word

- La mayoría de las Microcomputadoras maneja y almacena datos binarios en grupos de ocho bits.
  - 8 bits = 1 **byte**.
    - Un byte puede representar distintos tipos de datos o información.
- Frecuentemente los números binarios se agrupan en grupos de cuatro bits.
  - Un grupo de cuatro bits suele llamarse **nibble** (del inglés nibble = mordisquito).
- **Word** o **palabra** es un grupo de bits que representa una cierta unidad de información.
  - El **tamaño de palabra (Word size)** puede ser definido de distintas maneras en distintos sistemas. Lo más comun es definirlo como la cantidad de bits en la palabra que utiliza el sistema digital para operar.
  - La longitud de palabra en una PC actual es 64 bits.
  - La longitud de palabra en un 80x86 es 16 bits.
  - La longitud de palabra en un sistema basado en ARM es 32 bits.

Códigos alfanuméricos

- Representan caracteres, dígitos, símbolos y valores que pueden interpretarse como comandos o funciones.
- Normalmente se encuentran en el teclado de una computadora.
- Hace falta definir al menos 26 minúsculas, 26 Mayúsculas, 10 dígitos, 7 signos de puntuación y entre 20 y 40 caracteres adicionales.
- En distintos idiomas, puede ser necesario agregar más letras o símbolos
- ASCII : Está definido con 128 valores que pueden representarse con 7 bits.
- ASCII Extendido: Posee características que se adaptan a distintos idiomas y consta de 256 valores que pueden representarse con 8 bits.

Códigos alfanuméricos

ASCII

Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal	Character	HEX	Decimal
NUL (null)	0	0	Space	20	32	@	40	64	.	60	96
Start Heading	1	1	!	21	33	A	41	65	a	61	97
Start Text	2	2	“	22	34	B	42	66	b	62	98
End Text	3	3	#	23	35	C	43	67	c	63	99
End Transmit.	4	4	$	24	36	D	44	68	d	64	100
Enquiry	5	5	%	25	37	E	45	69	e	65	101
Acknowlege	6	6	&	26	38	F	46	70	f	66	102
Bell	7	7	`	27	39	G	47	71	g	67	103
Backspace	8	8	(	28	40	H	48	72	h	68	104
Horiz. Tab	9	9	)	29	41	I	49	73	i	69	105
Line Feed	A	10	*	2A	42	J	4A	74	j	6A	106
Vert. Tab	B	11	+	2B	43	K	4B	75	k	6B	107
Form Feed	C	12	,	2C	44	L	4C	76	l	6C	108
Carriage Return	D	13	-	2D	45	M	4D	77	m	6D	109
Shift Out	E	14	.	2E	46	N	4E	78	n	6E	110

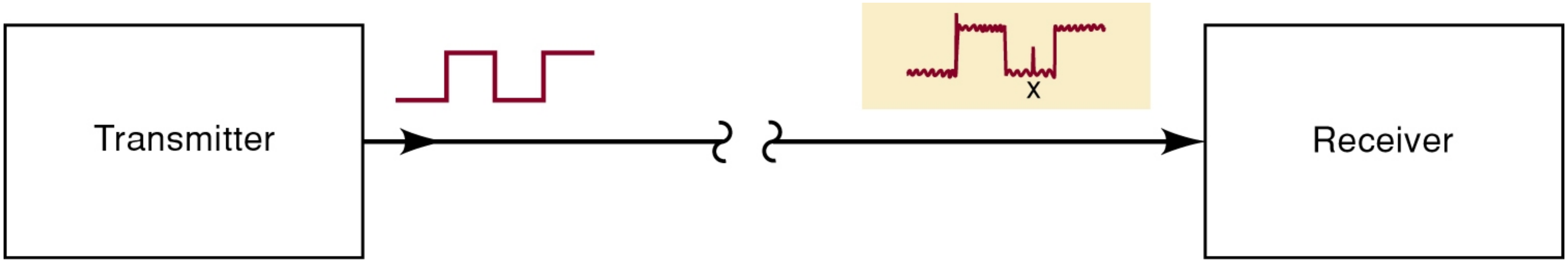
- Convertir la siguiente sentencia de C a codigo ASCII. Escribir en decimal, hexadecimal y binario

if (x>3)

Ascii	i	f		(	x	>	3	)
Hexade-cimal	*69*	*66*	*20*	*28*	*78*	*3E*	*33*	*29*
Decimal	*109*	*102*	*32*	*40*	*120*	*62*	*51*	*41*
Binario	*01101001*	*01100110*	*00100000*	*00101000*	*01111000*	*00111110*	*00110011*	*00101001*

2-9 Método de la Paridad para detectar errores

- El ruido eléctrico puede causar errores durante la transmisión.
  - Ruido? Fluctuaciones espurias y aleatorias en el voltaje o corriente. Aparecen en TODOS los sistemas electrónicos
  - NO puede eliminarse



- Muchos sistemas digitales emplean métodos para la detección e errores y algunas veces también su corrección.
- Uno de los sistemas de control de errores más simples y más utilizados es el método de PARIDAD.

Metodos de Paridad para detección de errores

- El método de paridad para la detección de errores requiere la adición de un bit extra al grupo de bits a transmitir.
  - Es llamado el bit de paridad.
  - Puede ser 0 o 1, dependiendo del número de 1s en el grupo de bits a transmitir..
- Hay dos métodos de paridad, PARIDAD PAR y PARIDAD IMPAR.
  - El transmisor y el receptor deben utilizar el mismo tipo de paridad.

Metodos de Paridad para detección de errores

- MÉTODO DE PARIDAD PAR:
- Se basa en completar el grupo de bits a transmitir con el bit de paridad de manera tal que la cantidad total de “1” que tenga la el conjunto sea PAR.
  - La cantidad de bits en 1 del (grupo de bits a transmitir + bit de paridad) debe ser par.
- Se cuenta la cantidad de “1” que tiene la palabra
- Si la cantidad es par, el bit de paridad se establece en 0
- Si la cantidad es impar el bit de paridad se establece en 1
- En ambos casos la cantidad TOTAL de bits en “1” debe ser par
- Ejemplo:

Grupo de bits a transmitir: 1000011

Bit de Paridad Par?

Metodos de Paridad para detección de errores

- MÉTODO DE PARIDAD IMPAR:
- Se basa en completar el grupo de bits a transmitir con el bit de paridad de manera tal que la cantidad total de “1” que tenga la el conjunto sea IMPAR.
  - La cantidad de bits en 1 del (grupo de bits a transmitir + bit de paridad) debe ser impar.
- Se cuenta la cantidad de “1” que tiene la palabra
- Si la cantidad es par, el bit de paridad se establece en 1
- Si la cantidad es impar el bit de paridad se establece en 0
- En ambos casos la cantidad TOTAL de bits en “1” debe ser IMPAR
- Ejemplo:

Grupo de bits a transmitir: 1000011

Bit de Paridad Impar?

Metodos de Paridad para detección de errores

- En ambos casos el bit de paridad se AGREGA al grupo original de bits a transmitir.
- El grupo de bits junto con la paridad, constituye la palabra de CÓDIGO.
- Por ejemplo si quiero transmitir palabras codificadas en ASCII con paridad PAR, la palabra de código tendrá 8 bits, 7 de los datos en ASCII y uno de paridad.

- Codificar HOLA MUNDO en ASCII
- Agregar los bits de paridad PAR (como MSB) correspondientes a cada símbolo ASCII.(PP)
- Repetir para paridad impar (PI)

H	O	L	A		M	U	N	D	O	
48	4F	4C	41	20	4D	55	4E	44	4F	ASCII
48	CF	CC	41	A0	4D	55	4E	44	CF	ASCII + PP
C8	4F	4C	C1	20	CD	D5	CE	C4	4F	ASCII + PI

Resumen parte I

- Repasamos los métodos de conversión entre bases numéricas
- Vimos un método de representación de digitos decimales llamado BCD.
- Vimos un sistema de detección de errores llamado paridad
  - Podemos utilizar dos tipos de control de paridad.
    - Paridad Par
    - Paridad Impar

Parte II

- Representación binaria con signo
  - Signo y Magnitud / Signo y Módulo (SM)
  - Complemento a 1 (CA1)
  - Complemento a 2 (CA2)
- Rangos de representación.
- Ventajas y desventajas entre los tres sistemas de representación.
- Adición, substracción multiplicación y división binarias en CA2.

Representación binaria con signo

- Para poder representar los números con signo utilizando solamente dos símbolos (0 y 1) es necesario codificar el signo del número:
  - Normalmente se codifica el signo con un 0 para números positivos y un 1 para números negativos
  - Se utilizan 3 diferentes sistemas de representación, que varían según las diferentes formas de codificar la magnitud:
    - Signo y magnitud (SM)
    - Complemento a 1 (CA1)
    - Complemento a 2 (CA2)

Representación binaria con signo

SIGNO Y MAGNITUD

- Dado que sólo es posible representar la **magnitud** con un número binario, el signo (+) o (-) puede mostrarse agregando un **bit de signo**.
  - El bit de signo 0 indica un número positivo
  - El bit de signo 1 indica un número negativo
  - Ejemplo:

$25_{10} = 11001_{BIN}$ (Binario Natural)

$+25_{10} = 011001_{SM}$ (Módulo y signo)

$-25_{10} = 111001_{SM}$ (Módulo y signo)

Representación binaria con signo

COMPLEMENTO a 1

- Si el número es positivo:
  - El MSB es un 0 (signo)
  - El resto de los bits son la magnitud en binario natural
- Si el número es negativo:
  - El MSB es un 1 (signo)
  - El resto de los bits son el complemento (a 1) de la magnitud

Ca1(A) = \A

La operación \A Se lee como "complemento a 1 de A" o más comunmente "A negado". Significa que se invierten todos los bits de A, incluyendo el de signo.

Equivalentemente, se puede definir como (siendo n el número de bits):

Ca1(A) *= 2^n –1 – A = (2^n −1)− A =11....11− A =* ***\A***

- Ejemplos:

$+25_{10}$ = 011001 $_{Ca1}$ (Complemento a 1)

-25_{10} = 100110 $_{Ca1}$ (Complemento a 1)

Representación binaria con signo

COMPLEMENTO a 2

- Si el número es positivo:
  - El MSB es un 0 (signo)
  - El resto de los bits son la magnitud en binario natural
- Si el número es negativo:
  - El MSB es un 1 (signo)
  - El resto de los bits son el complemento a 2 de la magnitud.
  - El complemento a dos de un número es su complemento a 1, + 1

Ca2(A) = Ca1(A)+1 = \A+1

Equivalentemente, se puede definir como(siendo n el número de bits):

Ca2(A) *= 2^n – A = 2^n − A + 1 – 1= (2^n −1)+1− A =11....11− A+1 =* ***\A+1***

Ejemplos:

$+25_{10} = 011001_{Ca2}$

$-25_{10} = 100111_{Ca2}$

Representación binaria con signo

COMPLEMENTO a 2

- No deben confundirse los siguientes conceptos
- “Operación de complementar a 2”:
  - ***Ca2(A) =   Ca1(A)+1 = \A+1***
- “Representación en complemento a 2”
  - Es el valor que representa un numero dado, que se obtiene siguiendo las siguientes reglas que permiten representar dos conjuntos de números:
    - Si el número es positivo, se utiliza la representación binaria natural, si el número se va a representar con n bits, el MSB debe ser “0”.
    - Si el numero a representar es negativo, se escribe su valor positivo y se aplica la operación de complementar a 2.
- La unión de ambos conjuntos es la representación en complemento a 2 utilizando n bits.
- La representación Complemento a 2 es la más utilizada en sistemas digitales.

Representación binaria con signo

COMPLEMENTO a 2

- Otra forma de realizar la operación de complementar a 2 es la siguiente:
- Comenzando por el LSB, copiar los bits hasta encontrar el primer 1 inclusive, e invertir el resto:
- EJEMPLO:

  Ca2(11000110)= 00111010

  Comprobación:

  Ca2(11000110) = 00111001+1= 00111010

Representación binaria con signo

Extensión del número de bits

- Un mismo número puede representarse con diferente número de bits:

$+25_{10} = 011001_{SM} = 00011001_{SM} = 0000000000011001_{SM}$

$+25_{10} = 011001_{CA1} = 00011001_{CA1} = 0000000000011001_{CA1}$

$+25_{10} = \underbrace{011001}_{\text{6 bits}}{}_{CA2} = \underbrace{00011001}_{\text{8 bits}}{}_{CA2} = \underbrace{0000000000011001}_{\text{16 bits}}{}_{CA2}$

- Si los números son positivos como en el ejemplo, en los tres sistemas se añaden ceros a la izquierda (en rojo) hasta completar el número de bits necesarios
- Nótese que el MSB se mantiene en 0 indicando el signo (positivo)

Extensión del número de bits

- En el caso de Números Negativos:
- Para representación SM, mantener el MSB que indica el signo y añadir ceros para completar el número de bits.
- Para representación Ca1 y Ca2 añadir unos a la izquierda del número hasta completar la cantidad de bits necesarios

$$-25_{10} = 111001_{SM} = 10011001_{SM} = 1000000000011001_{SM}$$

$$-25_{10} = 100110_{CA1} = 11100110_{CA1} = 1111111111100110_{CA1}$$

$$-25_{10} = \underbrace{100111}_{\text{6 bits}}{}_{CA2} = \underbrace{11100111}_{\text{8 bits}}{}_{CA2} = \underbrace{1111111111100111}_{\text{16 bits}}{}_{CA2}$$

Representación binaria con signo

Comparación:

- Vemos cómo la misma codificación en 3 bits representa distintos números en los tres sistemas
- Puede notarse que SM y Ca1 tienen dos representaciones del cero mientras que en Ca2 la representación del cero es única

Codificación	SM	Ca1	Ca2
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	-0	-3	-4
101	-1	-2	-3
110	-2	-1	-2
111	-3	-0	-1

Rangos de representación.

Cuál es el rango de valores que puede representarse con 8 bits?

- **Binario sin signo: $2^n = 2^8 = 256$ valores distintos (0 a 255)**

  $00000000 = 0_{10}$ al $11111111 = 255_{10}$

- **Magnitud y signo: $2^n - 1 = 2^8 - 1 = 255$ valores distintos**

  (–127 → – 0 y 0 → 127)

  $11111111 = -127_{10} \rightarrow 10000000 = -0_{10}$

  $00000000 = 0_{10} \rightarrow 01111111 = 127_{10}$

- **Ca1 : $2^n - 1 = 2^8 - 1 = 255$ valores distintos (–127 → –0 y 0 → 127)**

  $10000000_{Ca1} = -127_{10} \rightarrow 11111111_{Ca1} = -0_{10}$

  $00000000_{Ca1} = 0_{10} \rightarrow 01111111_{Ca1} = 127_{10}$

- **Ca2 : $2^n = 2^8 = 256$ valores distintos (–128 → 0 → 127)**

  $1000000_{Ca2} = -128_{10} \rightarrow 00000000_{Ca2} = 0_{10} \rightarrow 01111111_{Ca2} = 127_{10}$

Ventajas y desventajas de los distintos sistemas de representación.

- Módulo y Signo y Ca1 tienen dos representaciones del cero.
  - Esto ocasiona que tengo dos condiciones distintas para verificar un mismo número $\rightarrow$ aumenta la complejidad
- Las operaciones de suma y resta en módulo y signo son complicadas, en particular no se verifica que $x-z = x+(-z)$
- Ca2 tiene una unica representación del cero,esto me permite representar un valor más.
- En Ca1 y Ca2 se cumple que $x-z = x+(-z)$. Esto permite simplificar las unidades aritméticas

Operaciones en complemento a 2

SUMA

- Realizar la suma binaria normal de magnitudes.
  - Los bits de signo son sumados con los bits de magnitud.
- Si de la adición resulta un acarreo más alla del MSB, se descarta
  - Si el MSB es cero, entonces el resultado es positivo.
  - Si el MSB es uno el resultado es negativo y se encuentra representado en Ca2.
- **Overflow o desborde:**
  - Puede ocurrir sólo cuando se suman dos números positivos o negativos, y el resultado excede el rango de representación.
  - Cuando ocurre overflow, el signo del resultado es opuesto al signo de los operandos.

Suma en Ca2 de 5 bits. Ejemplos

Dos números positivos. (+9) + (+4) 0 1001 (+9) + 0 0100 (+4) 0 1101 (+13)	Número positivo mayor en módulo que número negativo. (+9) + (–4) 0 1001 (+9) + 1 1100 (–4) 1|0 0101 (+5)	Dos números iguales y de signo contrario. (+9) + (–9) 0 1001 (+9) + 1 0111 (–9) 1|0 0000 (0)
Dos números negativos (–9) + (–4) 1 0111 (–9) + 1 1100 (–4) 1|1 0011 (–13)	Número positivo menor en módulo que número negativo. (+4) + (–9) 0 0100 (+4) + 1 0111 (–9) 1 1011 (–5)	OVERFLOW (+9) + (+9) 0 1001 (+9) + 0 1001 (+9) 1 0010 (-14)

Operaciones en complemento a 2

RESTA.

- Se utiliza la propiedad $x – z = x + (–z)$
- Si se utiliza representación Ca2 de n bits, entonces:

$–z_{Ca2} = Ca2(z_{Ca2})$ dónde $Ca2(z)$ es la operación Ca2.

por lo tanto $x_{Ca2} – z_{Ca2} = x_{Ca2} + Ca2(z_{Ca2})$

o bien $x_{Ca2} – z_{Ca2} = x_{Ca2} + \backslash z_{Ca2} + 1$

($\backslash z_{Ca2}$ es la operación complemento a 1 o negación)

- En los sistemas digitales modernos se representan los números con signo en Ca2 y se implementa la resta a través de sumadores y negadores.

Operaciones en complemento a 2

MULTIPLICACIÓN

- Se realiza de manera similar a la multiplicación en base 10.
- En el caso de Números con signo representados en complemento a 2:
  - Si los dos números son positivos, se realiza la operación y se añade un bit de signo 0 (el resultado es positivo)
  - Si los dos números son negativos, se realiza la operación Ca2 sobre ambos números, se multiplican y al resultado se le añade un bit de signo 0 (el resultado es positivo).
  - Si uno de los números es negativo se hace la operación de Ca2 en ese número, se multiplican y al resultado se le vuelve a hacer la operación de Ca2 (el resultado es negativo). Si es necesario se extiende el signo.
  - Si los operandos están representados en n bits, el producto podrá llegar a necesitar 2n bits para su representación.

Operaciones en complemento a 2

MULTIPLICACIÓN (multiplicando y multiplicador, números positivos representados en 4 bits. Producto representado en 8 bits)

Ejemplo; 9 x 10

```
       1001  (9)
     x 1010 (10)
     -----------
       0000
      1001
     0000
    1001
    ------------
    01011010  (90)
```

Ejemplo; 1 x 1

```
       0001  (1)
     x 0001 (1)
     -----------
       0001
      0000
     0000
    0000
    ------------
    00000001  (1)
```

Ejemplo; 15 x 15

```
       1111 (15)
     x 1111 (15)
     -----------
       1111
      1111
     1111
    1111
    ------------
    11100001 (225)
```

Multiplicación en Ca2 de 4 bits

Ejemplo: un factor positivo y el otro negativo: –3 x 6

$-3=1101_{Ca2} \rightarrow +3=0011_{Ca2}$

```
      0011  (+3)
    x 0110  (+6)
____________________
      0000
     0011
    0011
   0000
____________________
  00010010  (+18)
```

Aplico:
Ca2(00010010)=
$= 11101110_{Ca2} = -18$

Ejemplo: dos factores negativos
(–1) x (–2)

$-1=1111_{Ca2} \rightarrow +1=0001_{Ca2}$

$-2=1110_{Ca2} \rightarrow +2=0010_{Ca2}$

```
      0001  (+1)
    x 0010  (+2)
____________________
      0000
     0001
    0000
   0000
____________________
  00000010  (+2)
```

DIVISIÓN o COCIENTE

- Tambien puede realizarse de la forma que se realiza para el sistema decimal
- Por ejemplo hagamos 25 / 4 y utilicemos 6 bits para representar los operandos.
- 25_{10} =011001 4_{10} =000100
- Dividendo → 11001 /100 → Divisor

```
 11001 /100
– 100   110 → Cociente
  01000
  –100
  00001
  – 000
  00001 → Resto
```

Se obtiene un cociente 6 y un resto 1

DIVISIÓN o COCIENTE

- Para las operaciones de división también valen las consideraciones que se tomaron en cuenta para la multiplicación respecto a los signos de los operandos y resultados
- Si dividendo y divisor son del mismo signo, entonces el cociente resultará positivo
- Si son de distinto signo, el cociente resultará negativo.
- Si el dividendo tiene m bits y el divisor tiene n bits, entonces el cociente tendrá (m-n) bits y el resto tendrá a lo sumo n bits.

Resumen parte II

- Se vieron tres formas de representar números binarios con signo:
  - Signo y Módulo (SM)
  - Complemento a 1 (Ca1)
  - Complemento a 2 (Ca2)

  Se enumeraron ventajas y desventajas de las distintas representaciones.
- Se vieron como se efectuan las operaciones básicas en el sistema más utilizado que es Ca2.