Representación de la información: Representación de enteros

Temario

- Enteros:
  - Sin signo *(SS)*.
  - Signo Magnitud *(SM)*.
  - Complemento a 2 *(C2)*.
  - Exceso.

- Es la forma en la que interpretamos un conjunto de *bits*.
- Solamente se puede operar entre datos representados con el mismo sistema de representación.
- El resultado de toda operación vuelve a estar representado en el mismo sistema.

Representación de números enteros positivos y el cero.

- Para representar cada número usamos su representación binaria, completando con ceros a la izquierda para utilizar todos los *bits*.
- Con $k\mathbf{b}$ podemos representar 2^k números distintos.
- Como queremos incluir el cero (y queremos representar todos los enteros en el rango) el último número no puede ser 2^k, sino $(2^k) - 1$.
- Por lo tanto el *Rango de Representación (RR)* de los enteros sin signo con $k\mathbf{b}$ es $[0, (2^k) - 1]$.

La suma de enteros sin signo en binario es muy similar a la suma en decimal. Solo hay que recordar las siguientes reglas:

La suma de enteros sin signo en binario es muy similar a la suma en decimal. Solo hay que recordar las siguientes reglas:

- Si no tenemos acarreo o el acarreo es cero:
  - $0 + 0 = 0$, y no generamos acarreo.
  - $1 + 0 = 1$, y no generamos acarreo.
  - $1 + 1 = 0$, y generamos 1 de acarreo.

La suma de enteros sin signo en binario es muy similar a la suma en decimal. Solo hay que recordar las siguientes reglas:

- Si no tenemos acarreo o el acarreo es cero:
  - $0 + 0 = 0$, y no generamos acarreo.
  - $1 + 0 = 1$, y no generamos acarreo.
  - $1 + 1 = 0$, y generamos 1 de acarreo.
- Si tenemos 1 de acarreo:
  - $(1) + 0 + 0 = 1$, y no generamos acarreo.
  - $(1) + 1 + 0 = 0$, y generamos 1 de acarreo.
  - $(1) + 1 + 1 = 1$, y generamos 1 de acarreo.

- El *bit* más significativo (el que esta más a la izquierda) representa el signo.
  - **0 → Positivo**
  - **1 → Negativo**
- Los *bits* restantes, el valor absoluto o magnitud.
- Si tenemos $k\mathbf{b}$, el RR es: $[-(2^{k-1}-1),(2^{k-1})-1]$

- El *bit* más significativo (el que esta más a la izquierda) representa el signo.
  - **0 → Positivo**
  - **1 → Negativo**
- Los *bits* restantes, el valor absoluto o magnitud.
- Si tenemos k**b**, el RR es: $[-(2^{k-1}-1),(2^{k-1})-1]$

*Ejemplo (con 8**b**):*

- $34_{10} = 0\,0100010$
- $-34_{10} = 1\,0100010$

Representación de enteros
(SM): Limitaciones

- Doble representación del cero.
- No utiliza eficientemente los $k\mathbf{b}$.
- No es conveniente para la aritmética (poco práctico).

Operación de Complemento a 2

- La operación de complementar a 2 consiste en obtener el opuesto de un número aritméticamente (el que tiene el mismo valor absoluto pero signo opuesto).
- $C2(a) + a = 0$

Operación de Complemento a 2

- La operación de complementar a 2 consiste en obtener el opuesto de un número aritméticamente (el que tiene el mismo valor absoluto pero signo opuesto).
- $C2(a) + a = 0$

Para obtener el complemento a 2 de un número:

- Se invierte cada uno de los bits (reemplazando 0 por 1 y viceversa)
- Al resultado se le suma 1.

Ejemplo: $C2(1010) = 01011$

Para representar un número a en complemento a 2 con k **bits**, comenzamos por considerar su signo:

Para representar un número a en complemento a 2 con k **bits**, comenzamos por considerar su signo:

- Si a es positivo o cero, lo representamos como en *SS(k)*, es decir, lo escribimos en base 2 con k**b**.
- Si a es negativo, tomamos su valor absoluto, lo representamos como en *SS(k)*, y le aplicamos la **operación** complemento a 2.

Ejemplo, representar 17 en $C2$ con $k = 8\mathbf{b}$:

Ejemplo, representar 17 en *C2* con $k = 8\mathbf{b}$:

- $17_{10} = 0001\,0001_2$ con $k = 8$ bits.

Ejemplo, representar 17 en *C2* con $k = 8\mathbf{b}$:

- $17_{10} = 0001\,0001_2$ con $k = 8$ bits.

Ejemplo, representar -17 en *C2* con $k = 8\mathbf{b}$:

Ejemplo, representar 17 en $C2$ con $k = 8\mathbf{b}$:

- $17_{10} = 0001\,0001_2$ con $k = 8$ bits.

Ejemplo, representar -17 en $C2$ con $k = 8\mathbf{b}$:

- Buscamos la representación de $|-17|$:
  $17 = 0001\,0001_2$ con $k = 8$ bits.

Ejemplo, representar 17 en *C2* con $k = 8\mathbf{b}$:

- $17_{10} = 0001\,0001_2$ con $k = 8$ bits.

Ejemplo, representar -17 en *C2* con $k = 8\mathbf{b}$:

- Buscamos la representación de $|-17|$:
  $17 = 0001\,0001_2$ con $k = 8$ bits.
- Aplicamos la operación de *C2* a la representación de $|-17|$:
  $C2(0001\,0001) = 1110\,1111 = -17_{10}$

El RR del sistema de representación complemento a dos, con $k\mathbf{b}$, es: $[-2^{k-1}, (2^{k-1}) - 1]$

- El cero tiene una única representación, lo que facilita las comparaciones.
- No requiere comprobaciones de signo antes de operar. Se opera directamente.
- El mecanismo de cálculo es eficiente y fácil de implementar en hardware.

Error MUY frecuente

Para representar cualquier número en *C2*... lo tengo que complementar a 2.... **NO!**

Error MUY frecuente

Para representar cualquier número en *C2*... lo tengo que complementar a 2.... **NO!** Al *representar* en *C2*, la *operación* de complementar a 2 únicamente se aplica cuando queremos **obtener el opuesto de un número**.

Una gran ventaja que aporta el sistema *C2* es que no es necesario implementar algoritmos de resta: Cuando se necesita efectuar una resta, se complementa el sustraendo y luego se lo suma al minuendo.

Las computadoras no restan: siempre suman.

Desbordamiento (overflow)

- Cuando el resultado de una operación aritmética excede la cantidad de dígitos del sistema, el resultado es inválido.
- Es importante que la computadora lo detecte, para informar al proceso, y que pueda descartar el resultado.

Desbordamiento en C2

- Puede ocurrir tanto al sumar dos números positivos como dos negativos.
- Es fácil detectarlo en este sistema, observando los dos últimos bits de acarreo (carry).

Detectar desbordamiento en C2

Se observan los dos últimos bits de acarreo (carry).

- Son diferentes? Hay overflow.
- Son iguales? No hay overflow.

Detectar desbordamiento en C2

Ejemplo: 23 - 9 (extender el signo para llegar a los k bits)

- 00010111 +
- 11110111

Detectar desbordamiento en C2

	1	1110 111
	23	0001 0111
+	-9	1111 0111
		0000 1110

Detectar desbordamiento en C2

Ejemplo: 123 + 9

- 01111011 +
- 00001001

Detectar desbordamiento en C2

$$
\begin{array}{r|l}
0 & 1111\ 011 \\
123 & 0111\ 1011 \\
+\quad 9 & 0000\ 1001 \\
\hline
 & 1000\ 0100
\end{array}
$$

Notación en exceso

- Se determina el intervalo de enteros a representar.
- Se determina la cantidad de bits necesarios para representar todos los enteros del intervalo.

Notación en exceso

- Un intervalo de $[a, b]$ enteros, comprende n valores, donde $n = b - a + 1$
- ¿Cuántos bits necesitamos para representar n valores?
- Por ejemplo, para un intervalo $[150, 157]$.

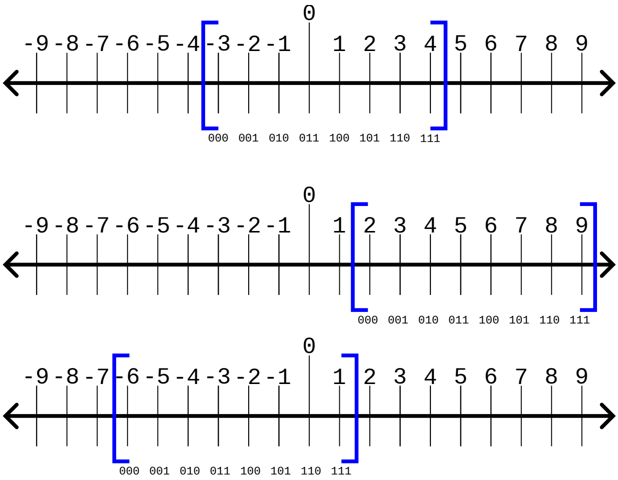
Notación en exceso

Representar un intervalo de enteros

Notemos que tanto a como b pueden ser negativos. Así podemos representar intervalos arbitrarios de enteros, lo que nos vuelve a dar un sistema de representación con signo.

Notación en exceso

Para saber qué número le corresponde a cada entero:
$valor\_decimal - a$



Temario

- Enteros:
  - Sin signo *(SS)*.
  - Signo Magnitud *(SM)*.
  - Complemento a 2 *(C2)*.
  - Exceso.

¿Consultas?