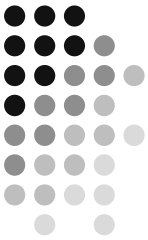
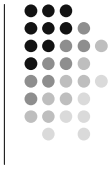
Elementos de Aritmética de Computadoras – Parte I



M. Vázquez, E. Todorovich, M. Tosini
Arquitectura I - Curso 2013
UNICEN

Cómputo Aritmético



- Las preguntas de fondo cuando se aborda el tema de la aritmética de computadoras son:
  - ¿Cómo represento los números?
  - ¿Qué algoritmos tengo para operar con esos números?
  - ¿Cuánto cuesta implementar y operar los algoritmos con las representaciones planteadas?
    - Área del dispositivo (en tecnologías digitales área de silicio)
    - Tiempo de cómputo
    - Energía por operación
  - Quizás la pregunta más importante es cuál es la aplicación en cuestión. ¿Qué requisitos tiene el problema?

Introducción a UAL

- Unidad Aritmético-Lógica (ALU: *Arithmetic Logic Unit*) es la parte del procesador que implementa las operaciones lógicas y aritméticas

- Las operaciones soportadas pueden ser
  - Ámbito de aplicación: generales o especializadas
  - Implementación: combinacionales o secuenciales
  - Paralelismo: serie o de dígito; paralelo o de vector
  - Operaciones: de desplazamiento, lógico o aritmético

Alcance del curso (UAL)

- Algoritmos de operaciones aritméticas e implementación
  - suma
  - resta
  - multiplicación
  - división
- Diferentes alternativas
  - con y sin signo
  - secuenciales y combinacionales
  - serie y paralelos
  - costos en tiempo y área
  - conceptos de *overflow*

Representación de números

- Enteros
  - Signo/Valor Absoluto
  - Complemento a base menos uno
  - Complemento a la base
  - Cero desplazado
- Fraccionarios
  - Punto fijo
  - Punto flotante IEEE 754-2008
    - Binario
    - Decimal

Racionales vs. Irracionales

Representación de números

(repaso Introducción Arquitectura de Sistemas)

- Signo/Valor Absoluto
  - Usado en representación de mantisa de números en punto flotante, los desplazamientos se realizan en forma directa
  - Simple negar un número
  - Tratamiento especial de signos en operaciones
- Cero desplazado
  - Flexible posicionamiento de frontera
  - Ideal para las comparaciones, usado en representación de exponentes de números en punto flotante
  - Se deben hacer correcciones en las operaciones

Representación de números
(repaso Introducción Arquitectura de Sistemas)

- Complemento base menos uno
  - Correcciones en las operaciones. Muy costosas en el caso de multiplicación
  - Simple negar un número
- Complemento base
  - No requiere correcciones en las operaciones aritméticas
  - Usado en representación de números enteros

Complemento a base dos

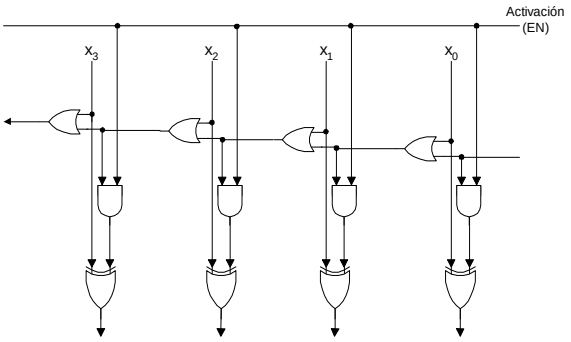
$$X = x_{n-1}\left(-2^{n-1}\right) + \sum_{i=0}^{n-2} x_i 2^i$$

- Rango de definición $[-2^{n-1}, 2^{n-1}-1]$
- Una sola representación para el cero.
- Cambio de signo

$$-X = \overline{X} + 1$$

Complemento a dos

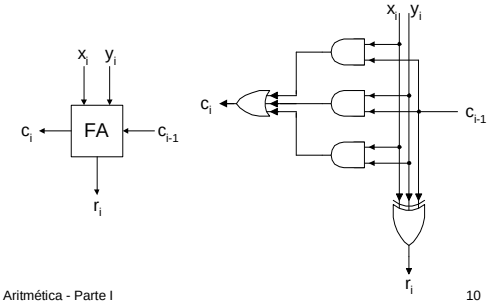
- El cambio de signo se puede hacer complementando todos los bits a la izquierda del último bit a '1' a la derecha.
- Ejemplo:
  - *X = 1011100 = (-36)$_{10}$*
  - *- X = 0100100 = (36)$_{10}$*



Sumador binario

- El sumador binario más clásico (*Ripple-Carry Adder*) utiliza en forma recursiva (en el tiempo o en el espacio) un sumador completo (*Full-Adder*; FA) de dos bits x_i, y_i más acarreo previo c_{i-1} que proporciona dos bits de resultados: la suma y el acarreo saliente c_i.

$$r_i = x_i \oplus y_i \oplus c_{i-1}$$
$$c_i = M(x_i, y_i, c_{i-1})$$

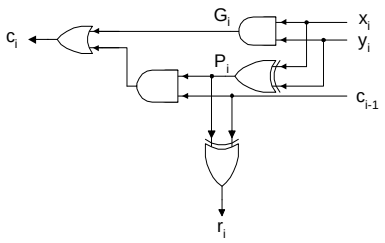


Sumador binario (*carry-chain*)

- El acarreo de salida se puede determinar a partir de las funciones de generación y propagación de acarreo.

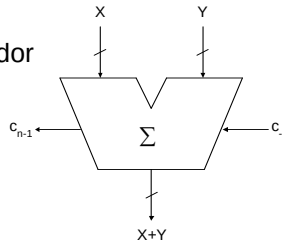
$$G_i = x_i \bullet y_i$$
$$P_i = x_i \oplus y_i$$

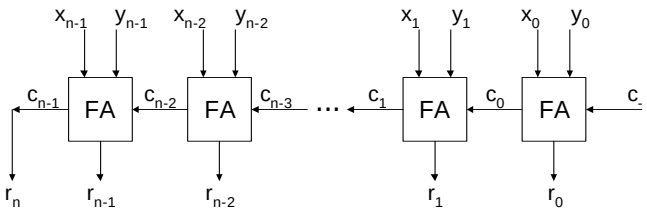
$$r_i = P_i \oplus c_{i-1}$$
$$c_i = G_i + P_i \bullet c_{i-1}$$



Sumador con propagación de acarreo (Ripple-Carry Adder)

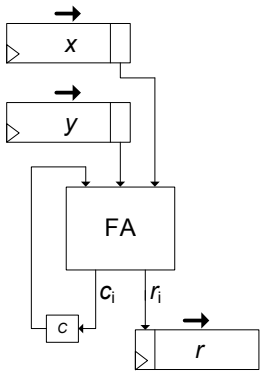
"Iteración" en el espacio del sumador completo. Combinacional





Sumador con propagación de acarreo (Ripple-carry adder)

"Iteración" en el tiempo. Serial, secuencial



Sumador binario con signo

- *X* ,*Y* son dos números binarios expresados en complemento a dos, la suma *R=X+Y*

$$\begin{array}{cccccccc} c_{n-1} & c_{n-2} & c_{n-3} & \dots & c_1 & c_0 & c_{-1} \\ & x_{n-1} & x_{n-2} & \dots & x_2 & x_1 & x_0 \\ & y_{n-1} & y_{n-2} & \dots & y_2 & y_1 & y_0 \\ \hline & r_{n-1} & r_{n-2} & \dots & r_2 & r_1 & r_0 \end{array}$$

Peso positivo, se usan FA comunes

c_{n-1} no se interpretará como un dígito más

x_{n-1}, y_{n-1} tienen peso negativo -2^{n-1} mientras que el acarreo c_{n-2} tendrá peso $+2^{n-1}$ con lo cual tiene que dedicarse un tratamiento especial al nivel *n-1*.

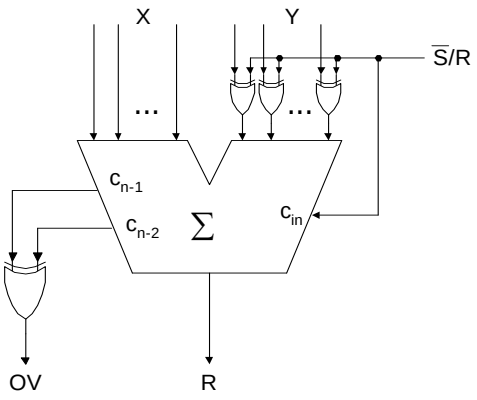
Sumador binario con signo

- Del análisis de x_{n-1}, y_{n-1}, c_{n-2} y c_{n-1} resultarán el *overflow* y del resultado de la suma

c_{n-1}	c_{n-2}	$x_{n-1} = y_{n-1} = 0$	$x_{n-1} = y_{n-1} = 1$	$x_{n-1} \neq y_{n-1}$
0	0	*X, Y >= 0* *R >= 0*		$r_{n-1} = 1$ *R < 0*
0	1	Overflow positivo		
1	0		Overflow negativo	
1	1		$r_{n-1} = 1$ *R < 0*	$r_{n-1} = 0$ *R >= 0*

Sumador binario con signo

- De la tabla queda: $V = c_{n-2} \oplus c_{n-1}$



Carry look-ahead adder

- Sumador con cálculo anticipado de acarreo (Carry look-ahead adder-CLA) .
- Utiliza funciones de generación y propagación de acarreo, G_i y P_i

$$G_i = x_i \bullet y_i$$
$$P_i = x_i \oplus y_i$$

Carry look-ahead adder

- Con estas definiciones el acarreo saliente se puede expresar como:

$$c_0 = G_0 \vee P_0 c_{-1}$$
$$\ldots$$
$$c_i = G_i \vee P_i c_{i-1}$$

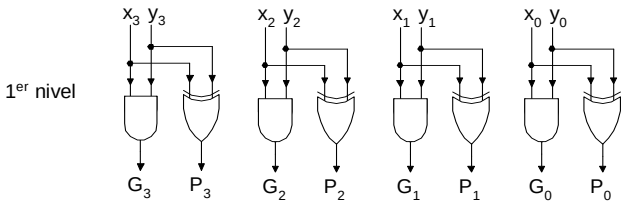
- Sustituyendo

$$c_0 = G_0 \vee P_0 c_{-1}$$
$$c_1 = G_1 \vee P_1 G_0 \vee P_1 P_0 c_{-1}$$
$$c_2 = G_2 \vee P_2 G_1 \vee P_2 P_1 G_0 \vee P_2 P_1 P_0 c_{-1}$$
$$\ldots$$

- Los c_k's se pueden calcular en 3 niveles lógicos, es decir:
  - Cálculo de G_i's, P_i's
  - Cálculo de productos (AND)
  - Suma (OR) de los productos

Carry look-ahead adder

Ejemplo: CLA de 4 bits



Carry look-ahead adder