Arquitectura de Computadoras para Ingeniería

(Cód. 7526)
1° Cuatrimestre 2018

Dra. Dana K. Urribarri
DCIC - UNS

Operaciones Aritméticas

Implementación de las operaciones aritméticas básicas:

1) Suma
2) Resta
3) Multiplicación
4) División

Multiplicadores

Multiplicadores

- Enteros
  - No signados
  - Signados

Multiplicación de enteros no signados

- Dos números de *n* dígitos en base *b*
  - Multiplicando: $M = m_{n-1}\ m_{n-2}\ \ldots\ m_0$
  - Multiplicador: $X = x_{n-1}\ x_{n-2}\ \ldots\ x_0 = \sum_{i=0}^{n-1} x_i\ b^i$
- El producto P (de 2*n* dígitos) será
  - $P = M\ X = M\ (\sum_{i=0}^{n-1} x_i\ b^i)$
- El producto P (de 2*n* dígitos) en binario
  - $P = M\ X = M\ (\sum_{i=0}^{n-1} x_i\ 2^i)$
- Se resuelve con *n* sumas de operandos de *n* bits. Por lo tanto, sería de $O(n^2)$

Algoritmo secuencial

- Parte de la expresión

$$P = M \cdot X = M \left(\sum_{i=0}^{n-1} x_i\, 2^i\right) = \sum_{i=0}^{n-1} M\, x_i\, 2^i$$

  Donde $x_i = 0$ o $x_i = 1$

- Arranca con el producto parcial en 0.
- Consiste de *n* pasos, donde el paso *j* multiplica el bit x_j por M y lo suma al producto parcial acumulado.

Enteros no signados

- El producto de dos enteros no signados de *n* bits, puede dar como resultado máximo P_{max}

$$(2^n - 1)(2^n - 1) = 2^{2n} \underbrace{- 2^{2n+1} + 1}_{<0} = 2^{2n-1} + \underbrace{(2^{2n-1} - 2^{n+1} + 1)}_{>0,\ n \geq 3}$$

- Luego

$$2^{2n-1} < P_{max} < 2^{2n}$$

- El resultado tiene como máximo 2*n* bits

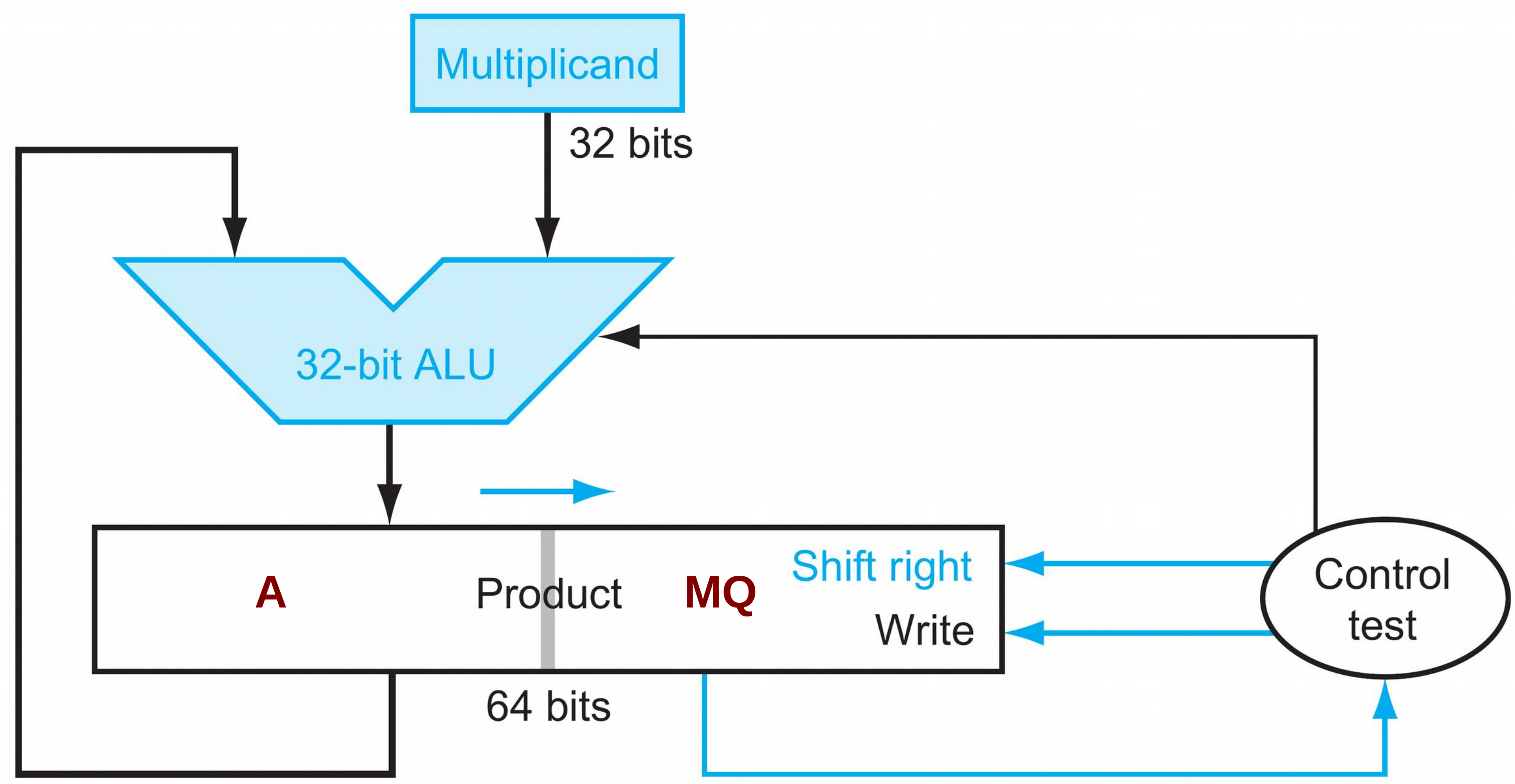
Algoritmo secuencial

- M = 6 = 0110
- X = 13 = 1101
- M X = ?

				0	1	1	0		
			×	1	1	0	1		Suma parcial:
				0	1	1	0		00000110
			0	0	0	0			00000110
		0	1	1	0				00011110
+	0	1	1	0					01001110
0	1	0	0	1	1	1	0		Producto

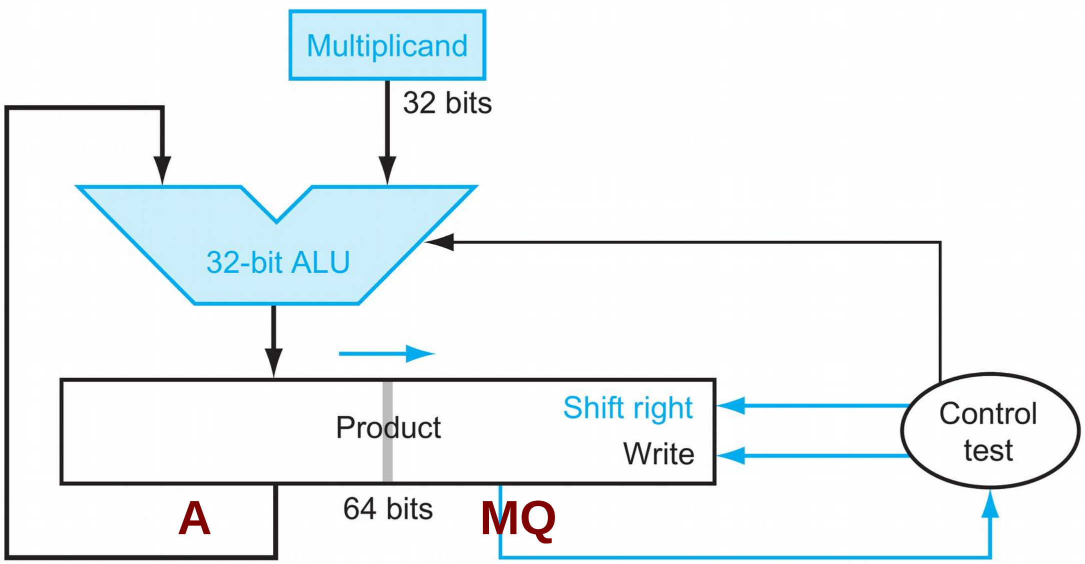
Hardware del algoritmo secuencial

- El registro Producto P es un registro doble A | MQ
  - Donde A y MQ son registros de *n* bits
- La ALU suma los registros de *n* bits A y Multiplicando



Hardware del algoritmo secuencial

- Inicio: Copiar el multiplicador en el registro MQ (parte derecha de P). Inicializar A (parte izq.) en 0.
- Repetir n veces
  - Paso 1: Si $P_0 = 0$, ir al paso 3
  - Paso 2: $A \leftarrow A$ + Multiplicando
  - Paso 3: Desplazar P 1 bit a derecha.



- El registro doble P contiene el resultado

Ejemplo

M = 6
X = 13
M X = ?

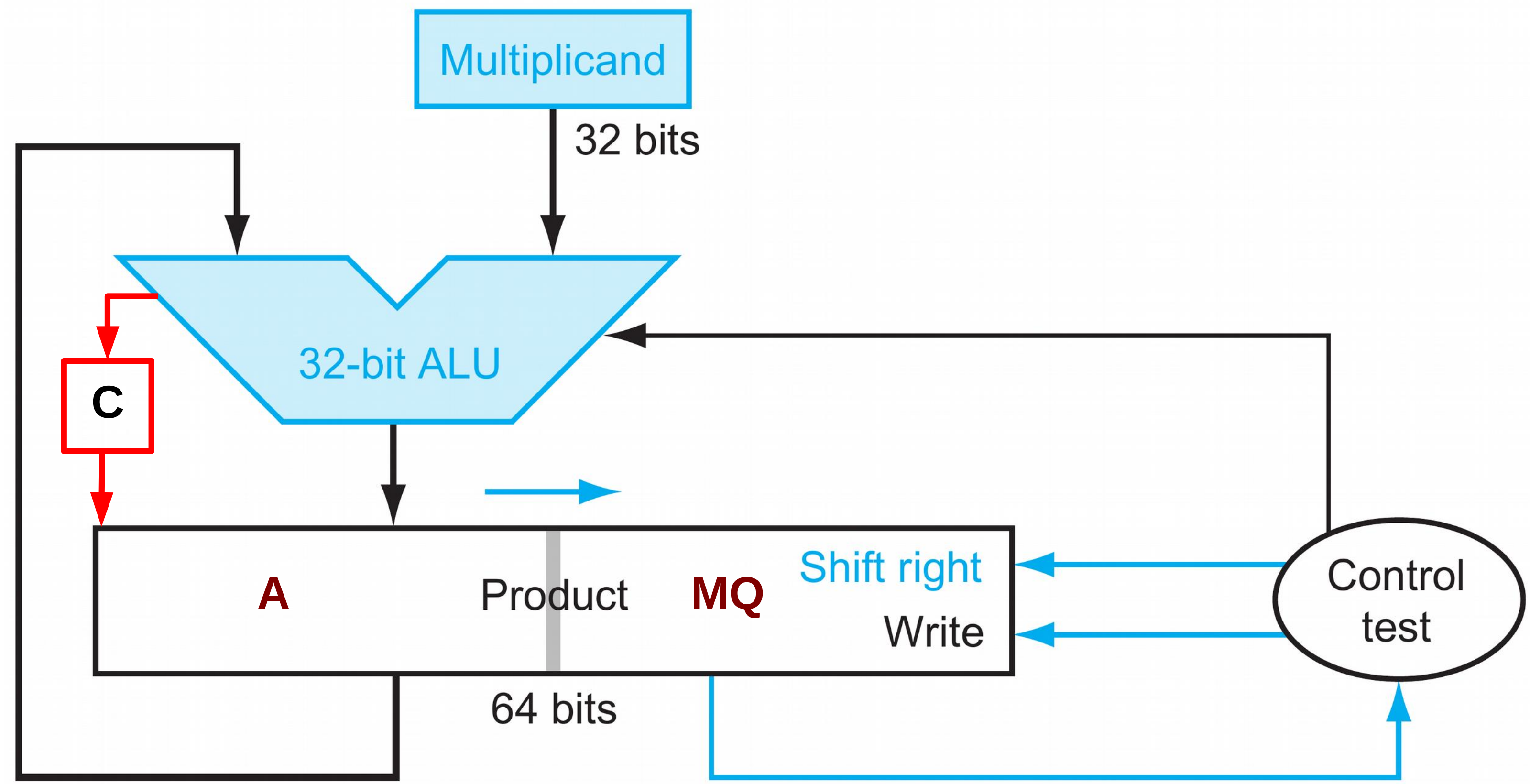
0	1	1	0	Multiplicando

Producto / Multiplicador

0	0	0	0	1	1	0	1	Test = 1 → **Sumar**
0	1	1	0	1	1	0	1	Shift
0	0	1	1	0	1	1	0	Test = 0 → **No Sumar** Shift
0	0	0	1	1	0	1	1	Test = 1 → **Sumar**
0	1	1	1	1	0	1	1	Shift
0	0	1	1	1	1	0	1	Test = 1 → **Sumar**
1	0	0	1	1	1	0	1	Shift
0	1	0	0	1	1	1	0	Resultado

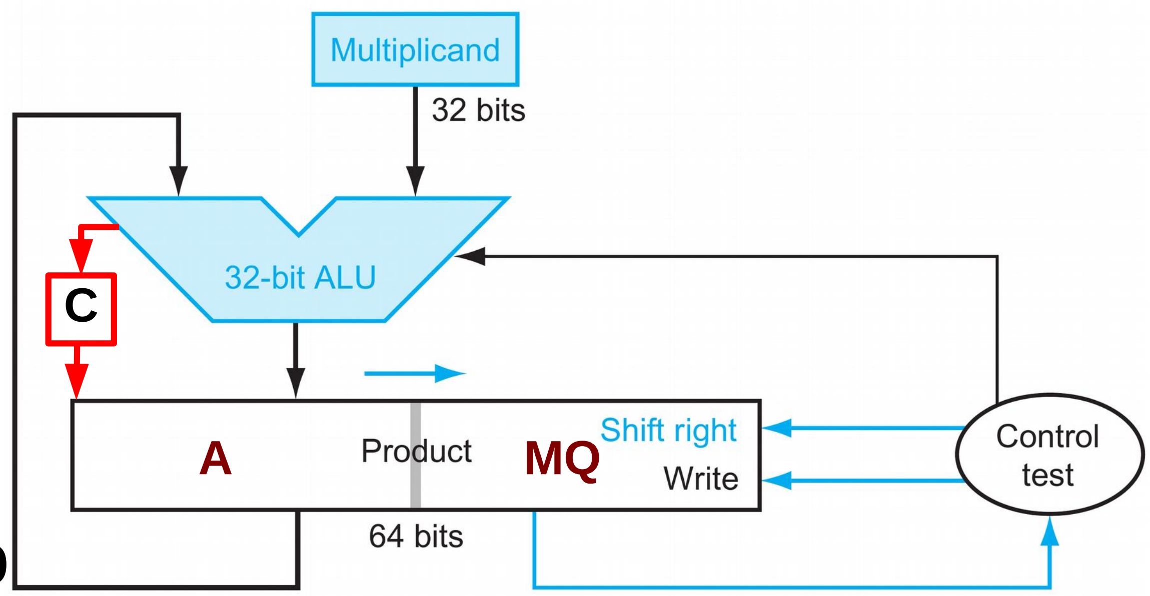
Hardware del algoritmo secuencial + carry

- Las sumas parciales pueden generar carry-out.
- Agregar el carry en el hw básico



Hardware del algoritmo secuencial

- Inicio: Copiar el multiplicador en el registro MQ. Inicializar A en 0.
- Repetir n veces
  - Paso 1:
    Si $P_0 = 0$, poner el carry en cero e ir al paso 3
  - Paso 2:
    $A \leftarrow A +$ Multiplicando
  - Paso 3: Desplazar P 1 bit a derecha incluyendo el carry.



Ejemplo

M = 13
X = 11
M X = ?

					1	1	0	1	Multiplicando

Carry	Producto				Multiplicador				
0	0	0	0	0	1	0	1	1	Test = 1 → **Sumar**
0	1	1	0	1	1	0	1	1	Shift
0	0	1	1	0	1	1	0	1	Test = 1 → **Sumar**
1	0	0	1	1	1	1	0	1	Shift
0	1	0	0	1	1	1	1	0	Test = 0 → **No Sumar** Shift
0	0	1	0	0	1	1	1	1	Test = 1 → **Sumar**
1	0	0	0	1	1	1	1	1	Shift
0	1	0	0	0	1	1	1	1	Resultado

Multiplicación de enteros signados

- Si los números están en signo-magnitud
  - Calcular el producto sin signo

    $|p| = |M| \cdot |X|$
  - Calcular el signo de forma separada

    $p_{2n-1} = y_{n-1} \oplus x_{n-1}$
- En el caso de complemento a 2 y complemento a uno.

  Distinguir entre multiplicando M negativo y multiplicador X negativo

Multiplicación de enteros signados

- Multiplicando negativo
  - $M = 2^n - |M|$
  - $P' = M \cdot X = (2^n - |M|) \cdot X = 2^n X - |M| \cdot X$
  - $P = - |M| \cdot X$

Multiplicación de enteros signados

- Solución 1:
  - La diferencia entre P y P' es 2^n X ∴ P = P' – 2^n X
  - Como P es un registro de 2*n* bits, – 2^n X en 2 complemento es $2^{2n} - 2^n$ X
  - Corregir el resultado del algoritmo restando X de la parte más significativa del registro P (esto no requiere ALU de 2*n* bits).

Multiplicación de enteros signados

- Solución 2:
  - Considerar A de doble precisión: $M = 2^{2n} – |M|$
  - $P' = M \cdot X = (2^{2n} – |M|) \cdot X = 2^{2n} X – |M| \cdot X$
  - 2^{2n} X es mayor que la longitud del registro P ∴ es carry que se descarta.
  - $P' = X \cdot M = 2^{2n} X – |M| \cdot X$
    $\equiv 2^{2n} – |M| \cdot X$
  - No implica ALU de 2*n* bits.

$$
\begin{array}{cccccccccccc}
 & & & & & 1 & 0 & 0 & 1 & 1 & (-13) \\
 & & & & \times & 0 & 1 & 0 & 1 & 1 & (+11) \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & & & & & \\
\hline
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & (-143)
\end{array}
$$

Multiplicación de enteros signados

- Multiplicando negativo: Solución 2 ✓
  - Si solo el multiplicando es negativo, no hay necesidad de cambiar el algoritmo.
  - Se suma un número negativo.
  - El hardware debe extenderse de forma tal que provea extensión de signo en el producto parcial.
    - Antes de la primera suma, en la extensión de signo ingresa 0.
    - Luego de la primera suma, en la extensión de signo ingresa m_{n-1}

Ejemplo: complemento a 2

$M = -5$
$m_{n-1} = 1$
$X = 3$
$M\,X = ?$

	1	0	1	1	Multiplicando				
	Producto				Multiplicador				
	0	0	0	0	0	0	1	1	Test = 1 → **Sumar**
+	1	0	1	1					
	1	0	1	1	0	0	1	1	Shift: extensión de signo
	1	1	0	1	1	0	0	1	Test = 1 → **Sumar**
+	1	0	1	1					
	1	0	0	0	1	0	0	1	Shift: extensión de signo
	1	1	0	0	0	1	0	0	Test = 0 → **No Sumar** Shift: extensión de signo
	1	1	1	0	0	0	1	0	Test = 0 → **No sumar** Shift: extensión de signo
	1	1	1	1	0	0	0	1	Resultado

Multiplicación de enteros signados

- Multiplicador negativo
  - $X = 2^n – |X|$
  - $P' = M \cdot X = M \cdot (2^n – |X|) = 2^n M – M \cdot |X|$
  - $P = – M \cdot |X|$

Multiplicación de enteros signados

- Solución 1:
  - La diferencia entre P y P' es 2^n M ∴ P = P' – 2^n M
  - Como P es un registro de 2*n* bits, – 2^n M en 2 complemento es $2^{2n} - 2^n$ M
  - Corregir el resultado del algoritmo restando M de la parte más significativa del registro P (esto no requiere ALU de 2*n* bits).

Multiplicación de enteros signados

- Solución 2:
  - Asumir X de doble precisión: $X = 2^{2n} – |X|$
  - $P' = M \cdot X = M \cdot (2^{2n} – |X|) = 2^{2n} M – M \cdot |X|$

    $\equiv 2^{2n} – M \cdot |X|$
  - ¿Problema?
    - Multiplicador de doble precisión implica el doble de iteraciones.

- Multiplicador Negativo: Solución 1 ✓

Ejemplo: complemento a 2

M = –7
m_{n-1} = 1
X = –5
M X= ?

1	0	0	1	Multiplicando

	Producto				Multiplicador				
	0	0	0	0	1	0	1	1	Test = 1 → **Sumar**
+	1	0	0	1					
	1	0	0	1	1	0	1	1	Shift: extensión de signo
	1	1	0	0	1	1	0	1	Test = 1 → **Sumar**
+	1	0	0	1					
	0	1	0	1	1	1	0	1	Shift: extensión de signo
	1	0	1	0	1	1	1	0	Test = 0 → **No sumar** Shift: extensión de signo
	1	1	0	1	0	1	1	1	Test = 1 → **Sumar**
+	1	0	0	1					
	0	1	1	0	0	1	1	1	Shift: extensión de signo
	1	0	1	1	0	0	1	1	Resultado. Corregir + 7
+	0	1	1	1					
	0	0	1	0	0	0	1	1	Resultado = 35

Ejemplo: complemento a 2

$M = -5$
$m_{n-1} = 1$
$X = -4$
$M\,X = ?$

	1	0	1	1	Multiplicando				
	Producto				Multiplicador				
	0	0	0	0	1	1	0	0	Test = 0 → **No Sumar** Shift: extensión de signo 0
	0	0	0	0	0	1	1	0	Test = 0 → **No Sumar** Shift: extensión de signo 0
	0	0	0	0	0	0	1	1	Test = 1 → **Sumar**
+	1	0	1	1					
	1	0	1	1	0	0	1	1	Shift: extensión de signo 1
	1	1	0	1	1	0	0	1	Test = 1 → **Sumar**
+	1	0	1	1					
	1	0	0	0	1	0	0	1	Shift: extensión de signo 1
	1	1	0	0	0	1	0	0	Resultado. Corregir + 5
+	0	1	0	1					
	0	0	0	1	0	1	0	0	Resultado = 20

Bibliografía

- *Capítulo 3 y 6. Computer Arithmetic Algorithms.* Israel Koren, 2da Edición, A K Peters, Natick, MA, 2002.

  Adapted from Koren, UMass. Copyright 2008 Koren, UMass and A.K. Peters.

- *Capítulo 10 y 11. Computer Arithmetic: Algorithms and Hardware Designs.* Behrooz Parhami, Oxford University Press, New York, 2002.

- *Apéndice J*. J. Hennessy & D. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann Publishers INC. 2011, 5ta Ed.

Suplementaria

- *Capítulo 42.* Editor Wai-Kai Chen. The VLSI Handbook. CRC Press. (2da Ed. 2007)