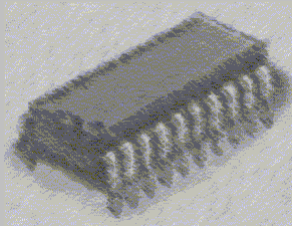


## Tema 3. La Unidad Aritmético-Lógica

*Arquitectura de  
Computadores*



I. T. Informática de Gestión

Curso 2009-2010

Tema 3:

Transparencia: 2 / 53

*La Unidad Aritmético-Lógica*

### Índice

- Estructura e implementación de la ALU
- Circuitos y algoritmos para la ALU:
  - Operadores lógicos y de desplazamiento
  - Operaciones sobre el signo
  - Suma en coma fija
  - Suma en coma flotante
  - Dígitos de guarda
  - Técnicas de redondeo
  - Multiplicación y división en coma fija
  - Multiplicación y división en coma flotante
- La ALU en la arquitectura von Neuman: camino de datos y unidades funcionales
- La unidad de ejecución del MC60888
- Bibliografía



Departamento de Automática  
Área de Arquitectura y Tecnología de Computadores

Arquitectura de Computadores  
I. T. Informática de Gestión

## Estructura e implementación de la ALU (I)

- **Unidad aritmético-lógica (ALU):** es el conjunto de operadores disponibles en un computador
- Formada por:
  - Operadores: aritméticos, lógicos y de desplazamiento
  - Registros para almacenar datos temporales
  - Registro de estado: conjunto de *flags* que indican situaciones ocurridas al operar
  - Registro contador de programa
  - Registro de direcciones de interrupción
- Tipos de ALU:
  - Coma fija
  - Coma flotante



## Estructura e implementación de la ALU (y II)

### Clasificación de los operadores:

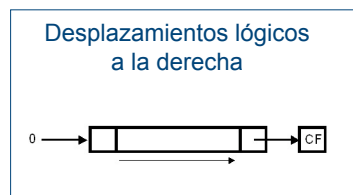
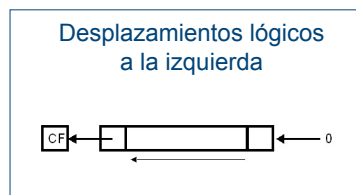
|                      |   |
|----------------------|---|
| Ámbito de aplicación | General<br>Especializado                  |
| Realización          | Combinacional<br>Secuencial               |
| Número de operandos  | Monádico<br>Diádico                       |
| Paralelismo          | Serie o de dígito<br>Paralelo o de vector |
| Operación            | De desplazamiento<br>Lógico<br>Aritmético |
| Tecnología empleada  | MOS<br>Bipolar                            |



## Circuitos y algoritmos para la ALU (I) Operaciones de desplazamiento (I)

- **Desplazamientos lógicos:**

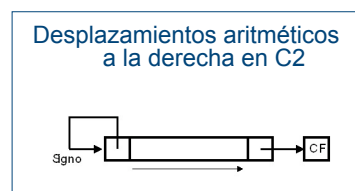
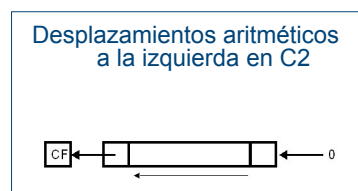
- Independientemente del sistema de representación de los operandos se introducen ceros por la derecha o por la izquierda según se trate de un desplazamiento a la izquierda o a la derecha, respectivamente
- El bit o los bits que salen suelen copiarse en el indicador de acarreo (el último que ha salido es el que queda)



## Circuitos y algoritmos para la ALU (II) Operaciones de desplazamiento (II)

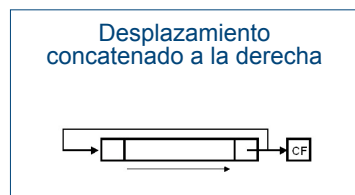
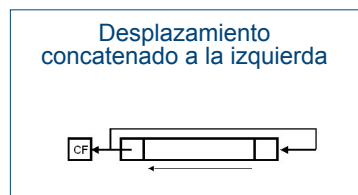
- **Desplazamientos aritméticos:**

- Equivalen a multiplicaciones y divisiones por dos, según sean hacia la izquierda o a la derecha, respectivamente.
- El sistema de representación de los operandos debe tenerse en cuenta si los operandos tienen signo
- El bit o los bits que salen suelen copiarse en el indicador de acarreo (el último que ha salido es el que queda)



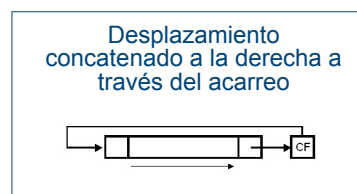
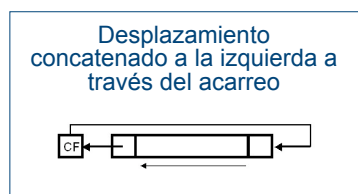
## Circuitos y algoritmos para la ALU (III) Operaciones de desplazamiento (III)

- **Desplazamientos circulares:**
  - Los bits que salen por un extremo entran por el otro
  - El bit o los bits que salen suelen copiarse en el indicador de acarreo (el último que ha salido es el que queda)



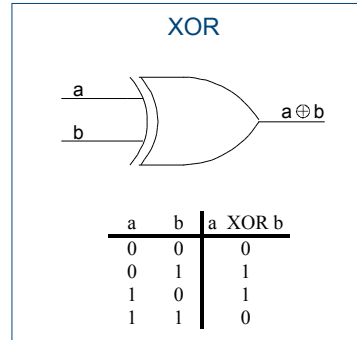
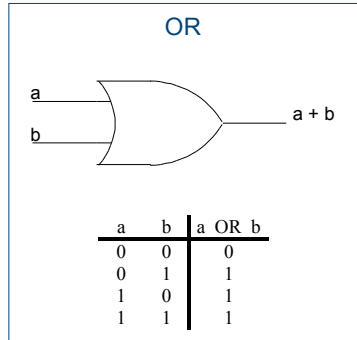
## Circuitos y algoritmos para la ALU (IV) Operaciones de desplazamiento (y IV)

- **Desplazamientos circulares a través del flag de acarreo:**
  - Los bits que salen por un extremo entran por el otro
  - El bit o los bits que salen suelen copiarse en el indicador de acarreo

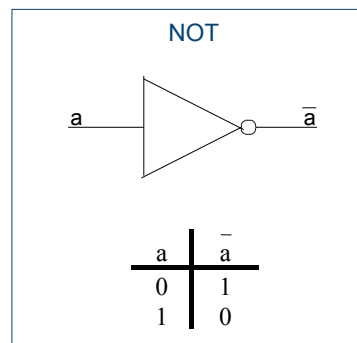
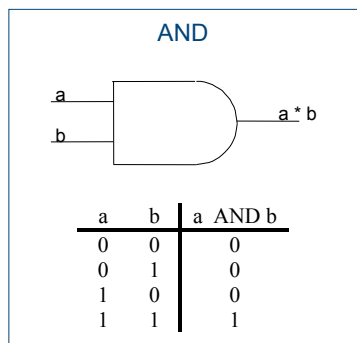


## Circuitos y algoritmos para la ALU (V) Operaciones Lógicas (I)

- Las operaciones lógicas realizan la operación sobre cada uno de los bits del operando o de los operandos

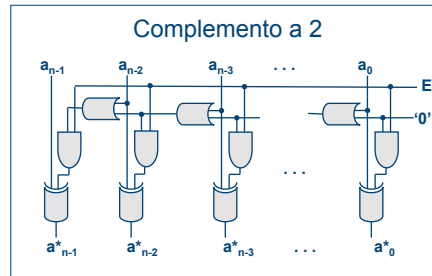
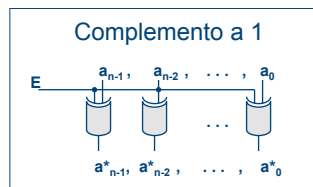
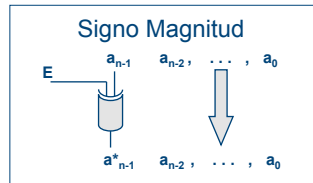


## Circuitos y algoritmos para la ALU (VI) Operaciones Lógicas (y II)



## Circuitos y algoritmos para la ALU (VII) Operaciones Aritméticas (I)

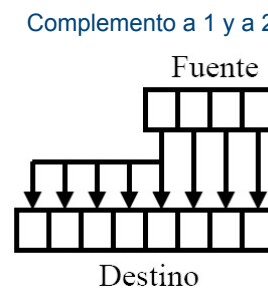
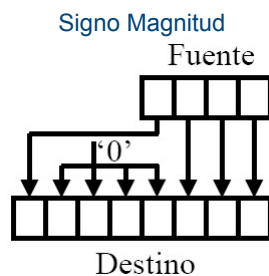
- Operaciones sobre el signo. Cambio de signo



## Circuitos y algoritmos para la ALU (VIII) Operaciones Aritméticas (II)

- Operaciones sobre el signo. Extensión de signo.

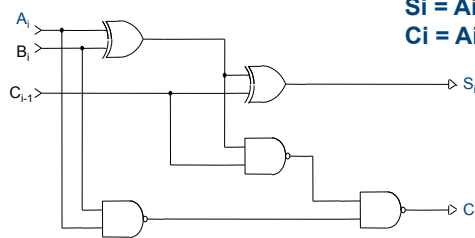
Es cuando se pasa de un operando de  $n$  bits a otro de  $m$  bits con  $m > n$ . Consiste en ver como "rellenar" los bits que sobran conservando el signo



## Circuitos y algoritmos para la ALU (IX) Operaciones Aritméticas (III)

- **Suma:** la suma es la operación más importante de todas, ya que:
  - Se emplea para el cálculo de la dirección de la siguiente instrucción
  - Se utiliza para el cálculo de las direcciones a los operandos
  - Otras operaciones la emplean: multiplicación, división

### Sumador elemental de un bit



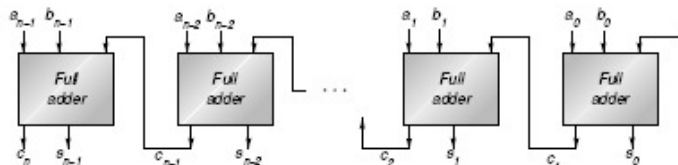
$$S_i = A_i \oplus B_i \oplus C_{i-1}$$

$$C_i = A_i B_i + B_i C_{i-1} + A_i C_{i-1}$$



## Circuitos y algoritmos para la ALU (X) Operaciones Aritméticas (IV)

- **Sumador con propagación de acarreo de  $n$  bits empleando sumadores completos (full adder) de 1 bit**



- **Problemas:**
  - Muy lento porque debe propagarse el acarreo desde el primer sumador al segundo, del segundo al tercero y así hasta que llegue al último sumador
  - El retardo es  $2 \cdot n \cdot r$  ya que es el máximo retardo de niveles de puertas que tiene que pasar para obtener el acarreo es de dos,  $n$  es el número de sumadores y  $r$  el retardo de las puertas

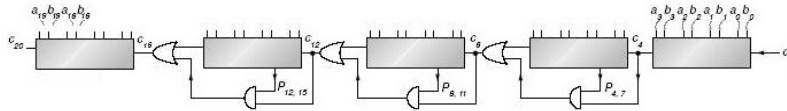






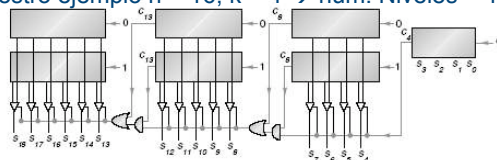
## Circuitos y algoritmos para la ALU (XIII) Operaciones Aritméticas (VII)

- **Aceleración de la suma. Salto de acarreo (CSK)**
  - Es un sumador intermedio entre uno de propagación y uno de anticipación.
  - Se basa en calcular las  $P_i$  que son de cálculo más sencillo.
- **Para un sumador de salto de acarreo de  $K$  bits:**
  - Primer propagador:  $2k + 1$  (el primer sumador un nivel más)
  - Número de puertas:  $2(n/k - 2)$
  - Último propagador:  $2k$
  - Total: número niveles =  $4k + 2n/k - 3$
  - En nuestro ejemplo  $n = 20, K = 4 \rightarrow$  num. Niveles = 21



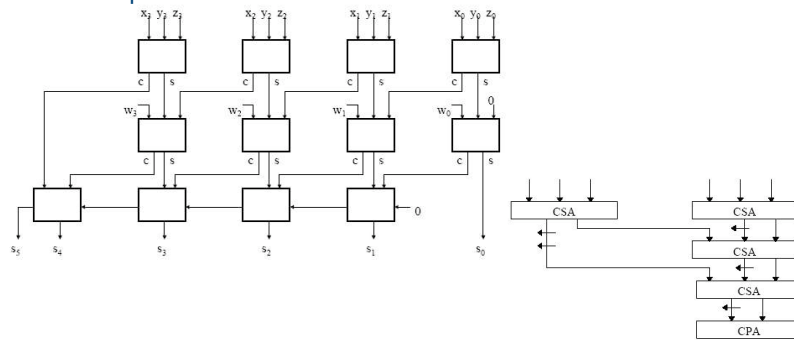
## Circuitos y algoritmos para la ALU (XIV) Operaciones Aritméticas (VIII)

- **Aceleración de la suma. Selección de acarreo**
  - Se basa en duplicar el hardware. Se suman los bits suponiendo acarreo cero y acarreo 1 y mediante multiplexores se elige el resultado correcto una vez conocido el acarreo
- **Para un sumador de  $n$  bits y bloques de  $k$  bits:**
  - Bloques:  $2k + 1$
  - Puertas:  $2(n/k - 2)$  (lógica de control del multiplexor)
  - Multiplexores:  $n/k - 1$  (una unidad de tiempo por multiplexor)
  - Total: número de niveles =  $2k + 2n/k$
  - En nuestro ejemplo  $n = 16, k = 4 \rightarrow$  num. Niveles = 16



## Circuitos y algoritmos para la ALU (XV) Operaciones Aritméticas (IX)

- **Aceleración de la suma: sumador sin propagación (CSA)**
  - Reduce el cálculo del tiempo casi a la mitad pero solamente es útil para más de 3 sumandos



## Circuitos y algoritmos para la ALU (XVI) Operaciones Aritméticas (X)

- **Tiempo de proceso y espacio requerido por cada uno de los distintos tipos de sumadores**

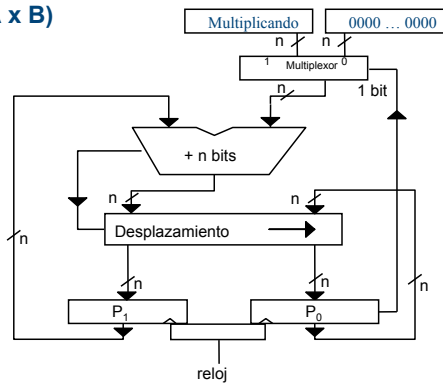
| Sumador      | Tiempo        | Espacio       |
|--------------|---------------|---------------|
| Propagación  | $O(n)$        | $O(n)$        |
| Anticipación | $O(\log n)$   | $O(n \log n)$ |
| Salto        | $O(\sqrt{n})$ | $O(n)$        |
| Selección    | $O(\sqrt{n})$ | $O(n)$        |



# Circuitos y algoritmos para la ALU (XVII)

## Operaciones Aritméticas (XI)

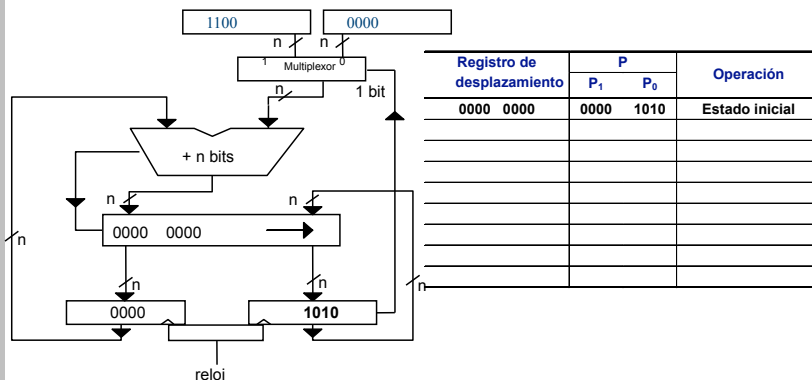
- Multiplicación. Algoritmo de suma desplazamiento (A x B)
- Inicialmente  $P_0 = B$
- Sólo números sin signo



# Circuitos y algoritmos para la ALU (XVIII)

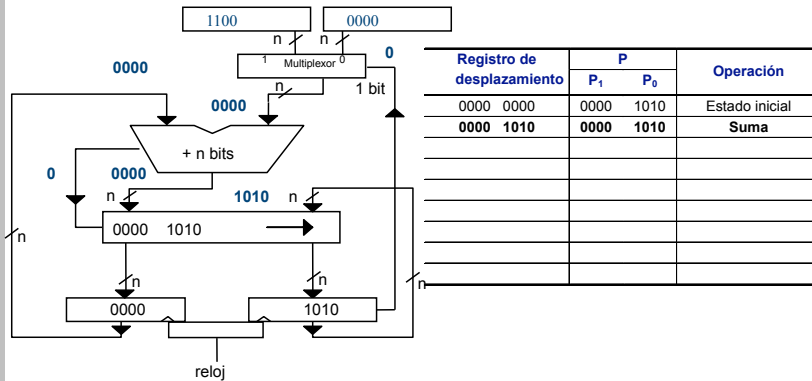
## Operaciones Aritméticas (XII)

- Ejemplo A = 1100 y B = 1010



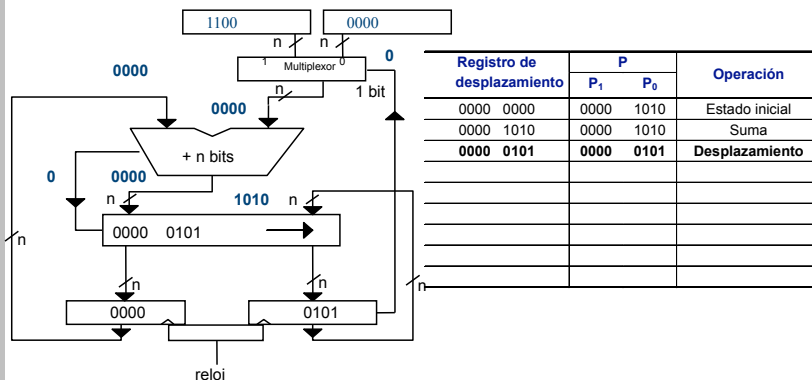
# Circuitos y algoritmos para la ALU (XIX) Operaciones Aritméticas (XIII)

- Ejemplo A = 1100 y B = 1010



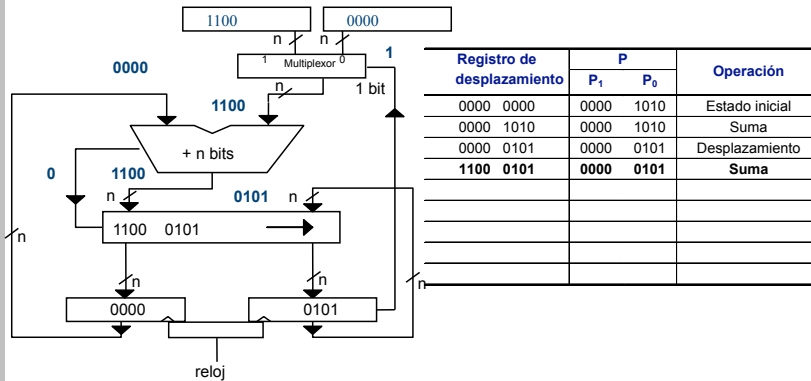
# Circuitos y algoritmos para la ALU (XX) Operaciones Aritméticas (XIV)

- Ejemplo A = 1100 y B = 1010



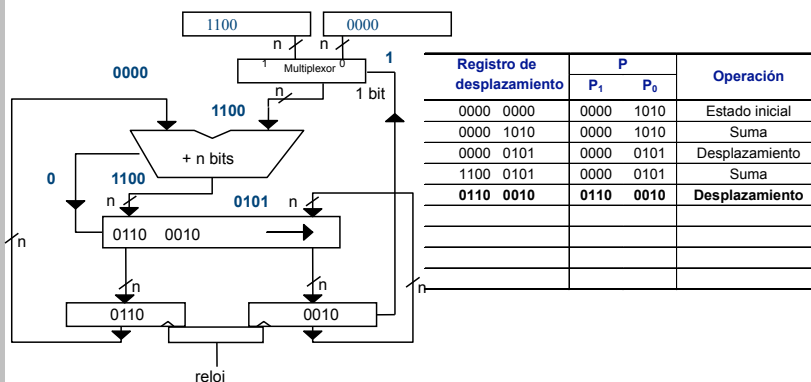
# Circuitos y algoritmos para la ALU (XXI) Operaciones Aritméticas (XV)

- Ejemplo A = 1100 y B = 1010



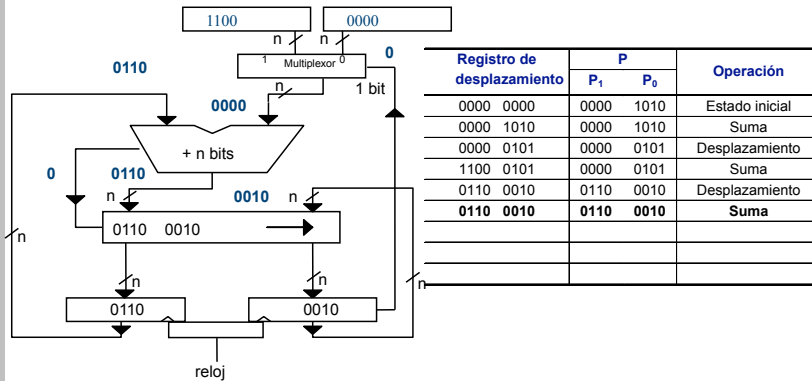
# Circuitos y algoritmos para la ALU (XXII) Operaciones Aritméticas (XVI)

- Ejemplo A = 1100 y B = 1010



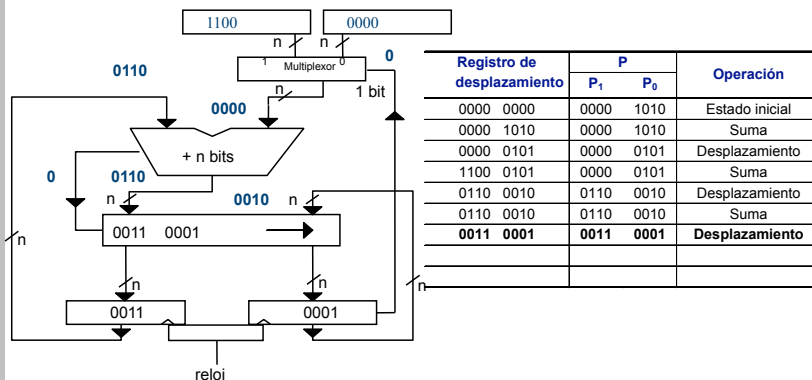
# Circuitos y algoritmos para la ALU (XXIII) Operaciones Aritméticas (XVII)

- Ejemplo A = 1100 y B = 1010



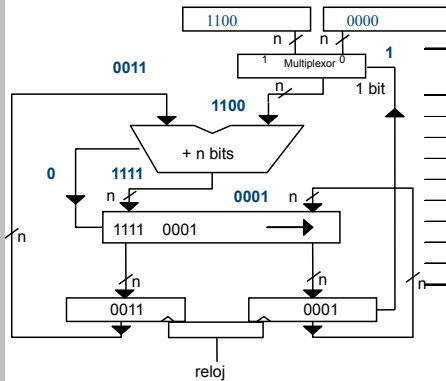
# Circuitos y algoritmos para la ALU (XXIV) Operaciones Aritméticas (XVIII)

- Ejemplo A = 1100 y B = 1010



# Circuitos y algoritmos para la ALU (XXV) Operaciones Aritméticas (XIX)

- Ejemplo A = 1100 y B = 1010

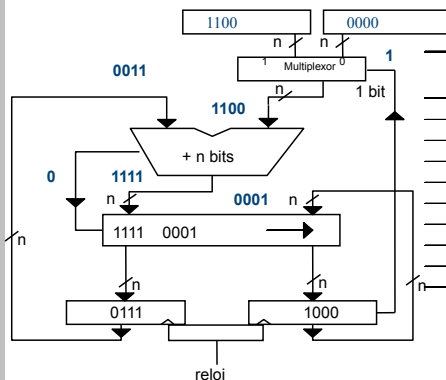


| Registro de desplazamiento | P              |                | Operación      |
|----------------------------|----------------|----------------|----------------|
|                            | P <sub>1</sub> | P <sub>0</sub> |                |
| 0000 0000                  | 0000           | 1010           | Estado inicial |
| 0000 1010                  | 0000           | 1010           | Suma           |
| 0000 0101                  | 0000           | 0101           | Desplazamiento |
| 1100 0101                  | 0000           | 0101           | Suma           |
| 0110 0010                  | 0110           | 0010           | Desplazamiento |
| 0110 0010                  | 0110           | 0010           | Suma           |
| 0011 0001                  | 0011           | 0001           | Desplazamiento |
| <b>1111 0001</b>           | <b>0011</b>    | <b>0001</b>    | <b>Suma</b>    |



# Circuitos y algoritmos para la ALU (XXVI) Operaciones Aritméticas (XX)

- Ejemplo A = 1100 y B = 1010



| Registro de desplazamiento | P              |                | Operación             |
|----------------------------|----------------|----------------|-----------------------|
|                            | P <sub>1</sub> | P <sub>0</sub> |                       |
| 0000 0000                  | 0000           | 1010           | Estado inicial        |
| 0000 1010                  | 0000           | 1010           | Suma                  |
| 0000 0101                  | 0000           | 0101           | Desplazamiento        |
| 1100 0101                  | 0000           | 0101           | Suma                  |
| 0110 0010                  | 0110           | 0010           | Desplazamiento        |
| 0110 0010                  | 0110           | 0010           | Suma                  |
| 0011 0001                  | 0011           | 0001           | Desplazamiento        |
| 1111 0001                  | 0011           | 0001           | Suma                  |
| <b>0111 1000</b>           | <b>0111</b>    | <b>1000</b>    | <b>Desplazamiento</b> |



## Circuitos y algoritmos para la ALU (XXVII) Operaciones Aritméticas (XXI)

- **Ajustes para operar con números representados en complemento a 1 y complemento a 2 (con signo)**
  - **Complemento a 2:** si el multiplicador es negativo, cuando llegue el 1 del bit más significativo al multiplexor, entonces, restar A
  - **Complemento a 1:** si el multiplicador es negativo, cuando llegue el 1 del bit más significativo al multiplexor, entonces, restar A. Y además, en la fase de inicialización P1 y el registro de desplazamiento toman el valor inicial del multiplicando



## Circuitos y algoritmos para la ALU (XXVIII) Operaciones Aritméticas (XXII)

- **Multiplicación. Algoritmo de Booth (C2)**
  - Se trata de evitar las sumas de cero que consumen ciclos de máquina y no aportan nada al cálculo puesto que solamente se desplazaría luego el resultado.
  - Emplea la recodificación del multiplicador buscando cadenas de 1's y afectándoles del peso de ese 1 ó del 0

- **A = 1011, B = 0110**
- **A x B = +Ax2<sup>3</sup> -Ax2<sup>1</sup>**

$$\begin{array}{r} 0001010 \\ 1011000 + \\ \hline 1100010 \end{array}$$

| X <sub>i</sub> | X <sub>i-1</sub> | significado               | acción asociada    | y <sub>i</sub> |
|----------------|------------------|---------------------------|--------------------|----------------|
| 0              | 0                | cadena de 0s              | desplazar          | 0              |
| 1              | 1                | cadena de 1s              | desplazar          | 0              |
| 1              | 0                | principio de cadena de 1s | restar y desplazar | -1             |
| 0              | 1                | fin de cadena de 1s       | sumar y desplazar  | 1              |





## Circuitos y algoritmos para la ALU (XXIX) Operaciones Aritméticas (XXIII)

- **División. Algoritmo de división con restauración (sin signo)**
- **Dividendo parcial inicial:** tomar tantos bits del dividendo como tenga el divisor.
- Garantizar que tanto el dividendo como el divisor son positivos (si empiezan por 1 añadirle un cero a la izquierda)
- Sumar al dividendo el complemento a 2 del divisor
  - Si el resultado es positivo:
    - Bajar un nuevo bit del dividendo
    - Añadir 1 al cociente
  - Si el resultado es negativo:
    - Sumar de nuevo el divisor
    - Bajar un nuevo bit del dividendo
    - Añadir un 0 al cociente
- Repetir hasta que no queden más bits para bajar del dividendo



## Circuitos y algoritmos para la ALU (XXX) Operaciones Aritméticas (XXIV)

- **División. Algoritmo de división con restauración (sin signo)**
  - A = 0100011, B = 0011, Cociente = 1011, Resto = 0010

$$\begin{array}{r}
 0\ 1\ 0\ 0\ 0\ 1\ 1 \quad | \ 0011 \\
 +1\ 1\ 0\ 1 \quad \downarrow \\
 \underline{1\ 0\ 0\ 0\ 1\ 0} \\
 +\ 1\ 1\ 0\ 1 \\
 \underline{1\ 1\ 1\ 1} \\
 0\ 0\ 1\ 0\ 1 \quad \leftarrow \text{Restauración} \\
 +\ 1\ 1\ 0\ 1 \\
 \underline{1\ 0\ 0\ 1\ 0\ 1} \\
 +\ 1\ 1\ 0\ 1 \\
 \underline{1\ 0\ 0\ 1\ 0}
 \end{array}$$



## Circuitos y algoritmos para la ALU (XXXI) Operaciones Aritméticas (XXV)

- **División. Algoritmo de división sin restauración (sin signo)**
- **Dividendo parcial inicial:** tomar tantos bits del dividendo como tenga el divisor.
- Garantizar que tanto el dividendo como el divisor son positivos (si empiezan por 1 añadirle un cero a la izquierda)
- Sumar al dividendo el complemento a 2 del divisor
  - Si el resultado es positivo:
    - Bajar un nuevo bit del dividendo
    - Añadir 1 al cociente
  - Si el resultado es negativo:
    - Sumar de nuevo el divisor en vez del complemento al bajar el siguiente bit
    - Bajar un nuevo bit del dividendo
    - Añadir un 0 al cociente
- Repetir hasta que no queden más bits para bajar del dividendo



## Circuitos y algoritmos para la ALU (XXXII) Operaciones Aritméticas (XXVI)

- **División. Algoritmo de división sin restauración (sin signo)**
  - A = 0100011, B = 0011, Cociente = 1011, Resto = 0010

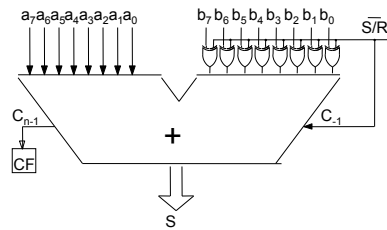
$$\begin{array}{r}
 0100011 \quad |0011 \\
 +1101 \quad \downarrow \\
 \hline
 00010 \\
 +1101 \quad \downarrow \\
 \hline
 11111 \\
 +0011 \quad \downarrow \\
 \hline
 \color{red}{1}00101 \\
 +1101 \\
 \hline
 \color{red}{1}0010
 \end{array}$$





# Circuitos y algoritmos para la ALU (XXXV) Circuitos (II)

- Sumador-restador en complemento a 2

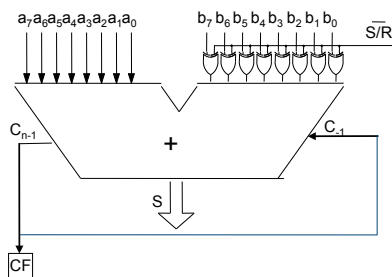


$$\text{Desbordamiento} = c_{n-1} \oplus c_{n-2}$$



# Circuitos y algoritmos para la ALU (XXXVI) Circuitos (III)

- Sumador-restador en complemento a 1



$$\text{Desbordamiento} = c_{n-1} \oplus c_{n-2}$$





## Circuitos y algoritmos para la ALU (XXXIX) Suma-resta en coma flotante (I)

- **Para sumar o restar números en coma flotante se debe:**
  1. Separar las mantisas de los exponentes
  2. Comparar los exponentes y:
    - Guardar el exponente mayor que será el del resultado salvo que el número salga desnormalizado
    - Restar del exponente mayor el menor. Dicho número será el número de veces que se tendrá que desplazar a la derecha la mantisa menor
  3. Desplazar la mantisa menor a la derecha para alinear las mantisas
  4. Realizar la suma o la resta
  5. Comprobar si el número está normalizado y en caso de que no lo esté, normalizarlo
  6. Realizar el redondeo si es preciso



## Circuitos y algoritmos para la ALU (XXXX) Suma-resta en coma flotante (y II)

- **Ejemplo de suma en coma flotante:**
  - Sean A y B dos números expresados en coma flotante. Exponente en exceso  $2^{n-1}$  con 8 bits y mantisa también con 8 bits, expresada en complemento a 2, normalizada y sin bit implícito
  - EA = 1000 0100    MA = 0100 0100
  - EB = 1000 0011    MB = 0111 0000
  - Comparamos exponentes EA(4) > EB(3). Exponente resultado EA
  - Desplazamos MB EA-EB veces, es decir,  $4 - 3 = 1$
  - 0100 0100
  - $\begin{array}{r} 0011\ 1000\ + \\ \hline 0111\ 1100 \end{array}$
  - ER = 1000 0100    MR = 0111 1100



## Circuitos y algoritmos para la ALU (XXXXI) Dígitos de guarda

- Los bits de guarda se añaden y se emplean únicamente dentro de la Unidad Aritmético-Lógica.
- Se emplean para aumentar la precisión de los resultados y permitir el redondeo y la normalización de manera correcta.
- Normalmente se emplean 2 bits de guarda y un bit retenedor que se añaden al final del número:

$$b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 \quad b_{g1} \quad b_{g2} \quad b_r$$

- Los bits  $b_8$  a  $b_0$  son un dato de 8 bits
- $b_{g1}$  es el primer bit de guarda que se emplea para la normalización
- $b_{g2}$  es el segundo bit de guarda que se emplea para el redondeo
- $b_r$  es el bit retenedor que se emplea para no perder la precisión en la operación de resta y que se mantiene a 1 al pasar un 1 por él



## Circuitos y algoritmos para la ALU (XXXXII) Técnicas de redondeo

- Al pasar de la ALU a los registros o la memoria, los bits de guarda y el bit retenedor, tienen que eliminarse por lo que se deberán tener en cuenta para mejorar la precisión del resultado
- Existen varias técnicas de redondeo entre las que destacan:
  - **Truncación:** se eliminan los bits de guarda y el bit retenedor
  - **Forzar a 1 el LSB:** se pone a 1 el bit menos significativo del resultado y se truncan los bits de guarda
  - **Redondeo al más próximo:** es la más difícil de implementar pero la que mejor resultado proporciona. Si supera la mitad del valor de los bits de guarda se suma uno al resultado. Si no llega a la mitad, se trunca. En el caso de que sea la mitad, se fuerza a par.
  - Para 3 bits de guarda las combinaciones 000, 001, 010, 011 truncan. Con 101, 110 y 111 se sumaría 1 al resultado. La combinación 100 sumará 1 si el LSB es 1 ó truncan si el LSB es 0



## ALU y Arquitectura Von Neumann (I) Camino de datos

- El órgano aritmético de la arquitectura von Neumann se denomina camino de datos o *datapath*
- Consta de: unidades aritmético-lógicas, desplazadores, registros y caminos de comunicación entre ellas
- El camino de datos contiene el estado del computador
- También contiene el registro contador de programa CP, el registro de direcciones de interrupción y el registro de estado
- El camino de datos influye en el coste del procesador, aproximadamente necesita la mitad de los transistores y la mitad del área de silicio del procesador
- La duración del ciclo de reloj está determinada por los circuitos más lentos por lo que en el procesador es el camino de datos el que hace de cuello de botella
- Los pasos clave para diseñar el camino de datos son:
  - Elegir el número de puertos del banco de registros
  - Seleccionar el tipo de ALU o de ALUs



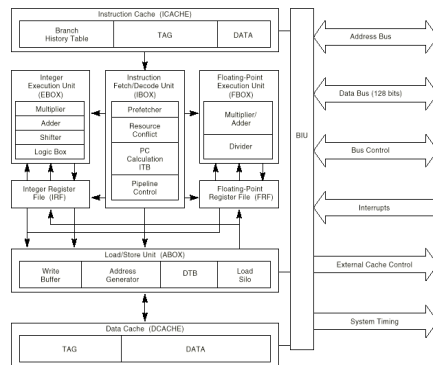
## ALU y Arquitectura Von Neumann (II) Camino de datos. Registro de estado

- Al realizar operaciones aritméticas, existen una serie de biestables de estado asociados a ellas y que se suelen recoger en un registro de estado
- El objetivo es guardar constancia de algunas características del resultado de las operaciones realizadas
- Los biestables de estado más frecuentes son:
  - Cero, signo, desbordamiento y acarreo
  - Paridad del resultado, acarreo BCD
- Condiciones de excepción. Algunas de ellas se basan en el contenido de los biestables de estado: desbordamiento.





## Ejemplo de hardware real (I) Alpha 21064. Aceleración de la suma (I)



### Unidad de cálculo entero (Ebox )

Contiene un camino de datos de 64 bits

- Adder
- Logic box
- Barrel shifter
- Bypasses
- Integer multiplier
- Archivo de registros con 32 registros de 64 bits, 4 ports de lectura y 2 de escritura.

### Unidad de cálculo de Punto Flotante (Fbox )

Contiene:

- Multiplicador/Sumador
- Divisor
- Un archivo de registros de 32 entradas, con registros de punto flotante de 64 bits
- Un registro de control accesible por el usuario (FPCR) que contine:
  - Controles para modos de redondeo dinámico
  - Información del flag de excepciones

Acepta una instrucción por ciclo, con excepción la instrucción de división.



## Ejemplo de hardware real (II) Alpha 21064. Aceleración de la suma (y II)

- El sumador de las unidades de enteros y de coma flotante del microprocesador Alpha 21064 es de 64bits y realiza 2 sumas por ciclo de reloj ya que es segmentado de 2 etapas
- Se combinan tres métodos de suma binaria:
  - Propagación de acarreo en bloques de 8 bits
  - Selección de acarreo en 2 bloques de 32 bits
  - Anticipación de acarreo entre ambos bloques de 32 bits
- En todos los caminos de datos se han colocado latches de manera que el operador cuenta con dos etapas aisladas que permiten realizar dos operaciones por ciclo



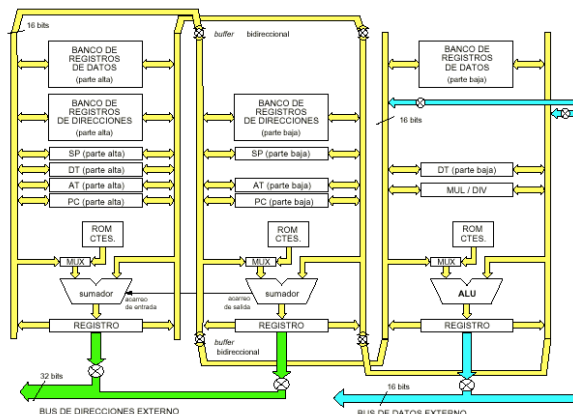
## Ejemplo de hardware real (III) La unidad de ejecución del MC68000 (I)



- 8 registros de direcciones de 32 bits divididos en parte alta y parte baja en 2 bancos separados
- Operaciones sobre:
  - Direcciones pueden ser de 16 o de 32 bits
  - Datos de 8, 16 ó 32 bits.
- El conjunto de los registros esta organizado en 3 secciones de modo que se puedan realizar operaciones a la vez en cada una de las secciones.
- La UE posee una unidad que realiza operaciones especiales



## Ejemplo de hardware real (y IV) La unidad de ejecución del MC68000 (y II)



## Bibliografía

- Estructura y diseño de computadores  
David A. Patterson y John L. Hennessy. Reverté, 2000  
Capítulo 4
- Arquitectura de computadores. Un enfoque cuantitativo  
John L. Hennessy y David A. Patterson. Mc Graw Hill, 3ª ed, 2002  
Apéndices A: aritmética de computadores
- Organización y arquitectura de computadores  
William Stallings. Prentice Hall. 1996  
Capítulo 8

