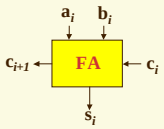
Unidades Aritméticas

Full Adder de un Bit

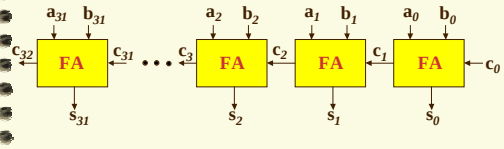


$s_i = a_i b_i c_i + a_i b_i c_i + a_i b_i c_i + a_i b_i c_i$

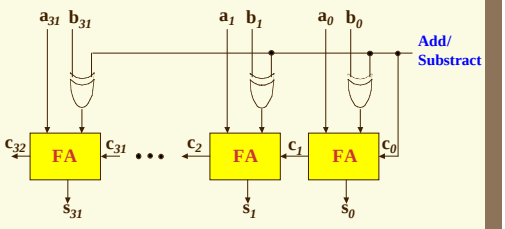
$c_{i+1} = a_i b_i + a_i c_i + b_i c_i$

a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder de 32 Bits



Sumador/Restador

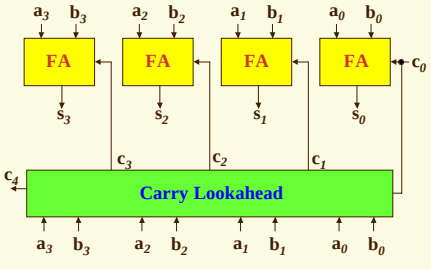


Carry Lookahead de 4 Bits

$P_i = a_i + b_i$ **(propagate function)**
$G_i = a_i b_i$ **(generate function)**

$C_1 = G_0 + P_0 c_0$
$C_2 = G_1 + P_1 G_0 + P_1 P_0 c_0$
$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_0$
$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$

Suma Rápida con Carry Lookahead



Multiplicación de Números Enteros

```
    1001  (9)
  x 0110  (6)
    0000
   1001
  1001
 0000
00110110  (54)
```


Add/Shift Algorithm

Para multiplicar dos números de n bits seguir el siguiente procedimiento:

1. Inicializar en cero un registro acumulador de 2n bits
2. Comenzando con el bit menos significativo del multiplicador realizar las siguientes dos operaciones para cada bit:

   Si el bit es igual a cero sumarle cero al acumulador y hacer un shift de una posición hacia la derecha del contenido del acumulador.

   Si el bit es igual a uno sumarle el multiplicando a los n bits más significativos del acumulador y hacer un shift de una posición hacia la derecha del contenido del acumulador (incluyendo el carry).

Ejemplo de Algoritmo de Multiplicación Add/Shift

```
1001 Multiplicando
x 0110 Multiplicador
                    Acumulador
Inicialmente        00000000
        Add         00000000
                    00000000
        Shift       00000000
        Add         10010000
                    10010000
        Shift       01001000
        Add         10010000
                    11011000
        Shift       01101100
        Add         00000000
                    01101100
        Shift       00110110
```


Algortimo Add/Shift Para Números con Signo (Two's Complement)

1. Inicializar en cero un registro acumulador de 2n bits
2. Comenzando con el bit menos significativo del multiplicador realizar las siguientes dos operaciones para cada bit excepto el bit n-1:

   Si el bit es igual a cero sumarle cero al acumulador y hacer un shift (sign extended) de una posición hacia la derecha del contenido del acumulador.

   Si el bit es igual a uno sumarle el multiplicando (sign extended) a los n+1 bits más significativos del acumulador y hacer un shift (sign extended) de una posición hacia la derecha del contenido del acumulador.

3. Para el bit n-1 realizar una de las siguientes dos operaciones:

   Si el bit es igual a cero sumarle cero al contenido del acumulador

   Si el bit es igual a uno sumarle el negativo del multiplicando al contenido del acumulador

Ejemplo de Algoritmo Add/Shift de Multiplicación con Signo

```
1001 Multiplicando
x 1110 Multiplicador
                    Acumulador
Inicialmente        00000000
        Add         00000000
                    00000000
        Shift       00000000
        Add         11001000
                    11001000
        Shift       11100100
        Add         11001000
                    10101100
        Shift       11010110
        Add         00111000
                    00001110
```


Algoritmo de Multiplicación Booth

El multiplicador se codifica como sigue:

m_i	m_{i-1}	Código	Operación
0	0	0	Suma cero
0	1	+1	Suma el multiplicando
1	0	-1	Suma el negativo del multiplicando
1	1	0	Suma cero

Se asume un cero antes del bit menos significativo del multiplicador

Ejemplo de Algoritmo Booth de Multiplicación

1001 Multiplicando
x 1010 Multiplicador

-1 +1 -1 0 Codificación del multiplicador

	Acumulador
Inicialmente	00000000
Add cero	00000000
	00000000
Add -multiplicando	00001110
	00001110
Add +multiplicando	11100100
	11110010
Add -multiplicando	00111000
	00101010

87

Algoritmo de Multiplicación Bit-Pairing Recoding

El multiplicador se codifica en pares de bits como sigue:

m_{i+1}	m_i	m_{i-1}	Código	Sumar al Acumulador
0	0	0	0	cero
0	0	1	+1	el multiplicando
0	1	0	+1	el multiplicando
0	1	1	+2	el multiplicando por dos
1	0	0	-2	el negativo del multiplicando por dos
1	0	1	-1	el negativo del multiplicando por uno
1	1	0	-1	el negativo del multiplicando por uno
1	1	1	0	cero

Se asume un cero antes del bit menos significativo del multiplicador y extensión de signo para los bits más significativos

88

Ejemplo de Algoritmo Bit-Pairing de Multiplicación

11001 Multiplicando
x 11010 Multiplicador

extensión de signo → 1110100 parejas de bits (cero implícito)

0 -1 -2 Codificación del multiplicador

	Acumulador
Inicialmente	00000000
Add -2 x multiplicando	00001110
	00001110
Add -multiplicando	00011100
	00101010
Add cero	00000000
	00101010

89

Ejercicios

Calcule los siguientes productos utilizando los algoritmos Add/Shift, Booth y Bit-Pairing

1011	110111010	0110100	1011110
x0101	x011100010	x1111010	x1100101

90

Algoritmo de División de Enteros

1. Inicializar a cero un registro A de n+1 bits, poner el dividendo en un registro Q y el divisor en un registro M
2. Realizar n veces lo siguiente:

2.1 Hacer un shift de A y Q una posición a la izquierda rellenado Q_0 con cero.

2.2 Sumarle el negativo de M a A.

2.3 Si A_n es 0 poner un 1 en Q_0. Si A_n es 1 sumarle M a A.

91

Ejemplo de División de Enteros

011 | 111

	A	Q	
Inicialmente	0000	111	
shift	0001	110	
add -M	1101	110	
	1110	110	ciclo uno
add M	0011	110	
	0001	110	
shift	0011	100	
add -M	1101	100	
	0000	100	ciclo dos
Set Q_0=1	0000	101	
shift	0001	010	
add -M	1101	010	
	1110	010	
add M	0011	010	ciclo tres
	0001	010	

92

Algoritmo de División "Nonrestoring"

1. Inicializar a cero un registro A de n+1 bits, poner el dividendo en un registro Q y el divisor en un registro M
2. Realizar n veces lo siguiente:
   - Si A_n es 0 :
     a) hacer un shift de A y Q una posición a la izquierda rellenado Q_0 con cero.
     b) sumarle el negativo de M a A
     c) si A_n es 0 poner un 1 en Q_0
   - Si A_n es 1 :
     a) hacer un shift de A y Q una posición a la izquierda rellenado Q_0 con cero.
     b) sumarle M a A
     c) si A_n es 0 poner un 1 en Q_0
3. Si A_n es 1 sumar M a A.

93

Ejemplo de División "Nonrestoring"

011 | 111

	A	Q	
Inicialmente	0000	111	
shift	0001	110	ciclo uno
add -M	1101	110	
	1110	110	
shift	1101	100	ciclo dos
add M	0011	100	
	0000	100	
Set Q_0=1	0000	101	
shift	0001	010	ciclo tres
add -M	1101	010	
	1110	010	
add M	0011	010	
	0001	010	

94

División de Números con Signos

- ✓ Los números se convierten a números positivos
- ✓ Se calcula el cociente y el residuo
- ✓ Se convierte el cociente a número positivo o negativo dependiendo de los signos del dividendo y el divisor
- ✓ Se le asigna al residuo el signo del dividendo

95

Ejercicios de División

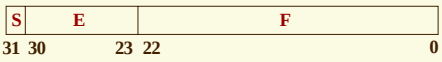
- ✓ Para cada uno de los siguiente casos aplique los algoritmos de división de enteros y "nonrestoring".

1000 | 1101 11111 | 00100 00010 | 11000

96

Floating Point Arithmetic: IEEE 754 Foalting Point Standard

Single Precision



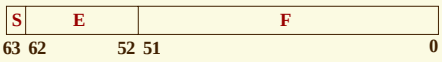
Number = S 1.F x 2^{E-127}

S - sign (0 = + , 1 = -)
E - exponent
F - fraction

97

Floating Point Arithmetic: IEEE 754 Foalting Point Standard

Double Precision



Number = S 1.F x 2^{E-1023}

98

Operations of the IEEE 754 Standard

- ✓ Addition
- ✓ Substraction
- ✓ Multiplication
- ✓ Division
- ✓ Remainder
- ✓ Square Root
- ✓ Binary to decimal conversion
- ✓ Decimal to binary conversion

99

Exceptions of the IEEE 754 Standard

- ✓ Invalid Operation
- ✓ División by 0
- ✓ Overflow
- ✓ Underflow
- ✓ Inexact

100

Conversión de Punto Flotante a Decimal (Precisión Sencilla)

1 00100100 10110000000000000000000

$N = -1.6875 \times 2^{-91}$

0 10100100 00000100000000000000000

$N = +1.015625 \times 2^{+37}$

0 10000101 10000000000000000000000

$N = +1.5 \times 2^{+6} = 96$

0 00000000 00000000000000000000000

$N = 0$

101

Algoritmo de Conversión de Decimal a Punto Flotante (Precisión Sencilla)

1. Representar el número como una suma de cantidades 2^a.
   Ejemplo: $225 = 128 + 64 + 32 + 1 = 2^{+7} + 2^{+6} + 2^{+5} + 2^{+0}$
2. Tomar el exponente del término mayor como el exponente del número.
3. Dividir los restantes términos por el término mayor

   $\frac{2^{+6}}{2^{+7}} + \frac{2^{+5}}{2^{+7}} + \frac{2^{+0}}{2^{+7}} = 2^{-1} + 2^{-2} + 2^{-7}$
4. Poner un uno en las posiciónes de la fracción que corresponden al peso de cada uno de los términos restantes

$225 =$ 0 10000110 11000010000000000000000

102

Conversión de Decimal a Punto Flotante (Precisión Sencilla)

$N = 63.75$

0 10000100 11111110000000000000000

$N = 521$

0 10001000 00000100100000000000000

$N = -.75$

1 01111110 10000000000000000000000

103

Ejercicios

- ✓ Convertir los siguientes números a su equivalente decimal

  0 10111100 11100100000000000000000

  0 11101110 00110100100000000000000

  1 00000110 10000010000000000000101
- ✓ Convertir los siguientes números a su equivalente de punto flotante

  55, 245, -129, -77.125

104

Normalización

Mantisa sin Normalizar

0.00101100000000000000000

implícitos

Mantisa Normalizada

1.01100000000000000000000

El exponente final es igual al exponente parcial menos el número de "shifts" necesarios para normalizar el número.

105

Procedimiento para Sumar o Restar Números de Punto Flotante

1. Escoger el número con el exponente menor y hacer un "shift" hacia la derecha de la mantisa (incluir el uno implícito) un número de posiciones igual a la diferencia entre los exponentes.
2. Hacer el exponente del resultado igual al exponente mayor.
3. Realizar la suma o resta de las mantisas y determinar el signo del resultado
4. Normalizar el resultado de ser necesario.

106

Ejemplo de Resta

0 00100011 00001000000000000000000
- 0 00100000 11000000000000000000000

1.00001000000000000000000
- 0.00111000000000000000000 shift
0.11010000000000000000000 resta
1.10100000000000000000000 normalizar

Resultado

0 00100010 10100000000000000000000

107

Procedimiento para Multiplicación de Números de Punto Flotante

1. Sumar los exponentes y restarle 127
2. Multiplicar las mantisas y determinar el signo del resultado
3. Normalizar el resultado

108

Procedimiento para División de Números de Punto Flotante

1. Restar los exponentes y sumarle 127
2. Dividir las mantisas y determinar el signo del resultado
3. Normalizas el resultado de ser necesario

109

Redondeo y "Guard Bits"

- ✓ IEEE standard utiliza tres bits (guard bits) para redondear los resultados de operaciones
- ✓ Estos bits son eventualmente truncados

110