**UNIVERSIDAD AUTONOMA** **DE BAJA CALIFORNIA**

**Organización de computadoras y lenguaje ensamblador**

**Practica 3.** Diseño de una Unidad Aritmética y Lógica

**Alumno**

Caudillo Sánchez Diego

**Matricula**

1249199

**Grupo**

551

**Docente**

Dr. Mauricio Alonso Sánchez

**Fecha de entrega**

08/Marzo/2019

**Objetivo**

Diseñar un ALU de 8 bits

**Materiales**

Logisim

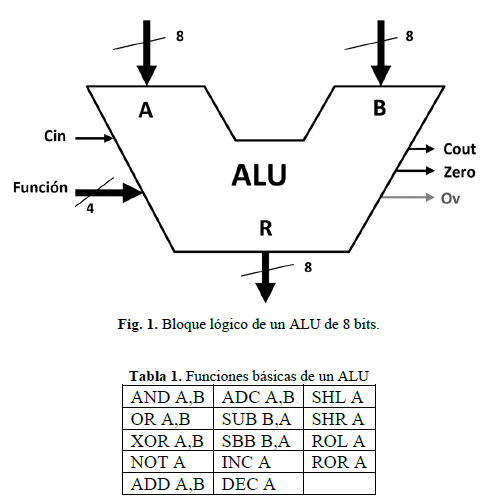
**Teoría**

Hacer una reseña sobre:

* ALU
* FPU
* Describir cada entrada y salida del ALU de la Figura 1.
* Describir cada función básica de la Tabla 1.

**Desarrollo**

Diseñar un ALU, ver la figura 1, que ejecute las funciones incluidas en la tabla 1.



**ALU**

Es un contador digital capaz de realizar las operaciones aritméticas y lógicas entre los datos de un circuito; suma, resta, multiplica y divide, así como establece comparaciones lógicas a través de los condicionales lógicos “si”, “no”, y, “o”. Desde los circuitos más simples, como relojes y calculadoras, hasta complejos circuitos, como los microchips actuales, todos incluyen al menos una Unidad aritmético-lógica, que varía su poder y complejidad según su finalidad.

**FPU**

Es un componente del CPU especializado en el cálculo de operaciones en punto flotante. Las operaciones básicas que toda FPU puede realizar son las aritméticas (suma y multiplicación), si bien algunos sistemas más complejos son capaces también de realizar cálculos trigonométricos y/o exponenciales.

No todas los CPU’s tienen una FPU dedicada. En ausencia de FPU, el CPU puede utilizar programas en microprogramas para emular una función en coma flotante a través de la unidad aritmético-lógica (ALU), la cual reduce el coste del hardware a cambio de una sensible pérdida de velocidad.  
En algunas arquitecturas, las operaciones de punto flotante se tratan de forma completamente distinta a las operaciones enteras, con registros dedicados y tiempo de ciclo diferentes. Incluso para operaciones complejas, como la división, podrían tener un circuito dedicado a dicha operación.

**Entradas y salidas ALU**

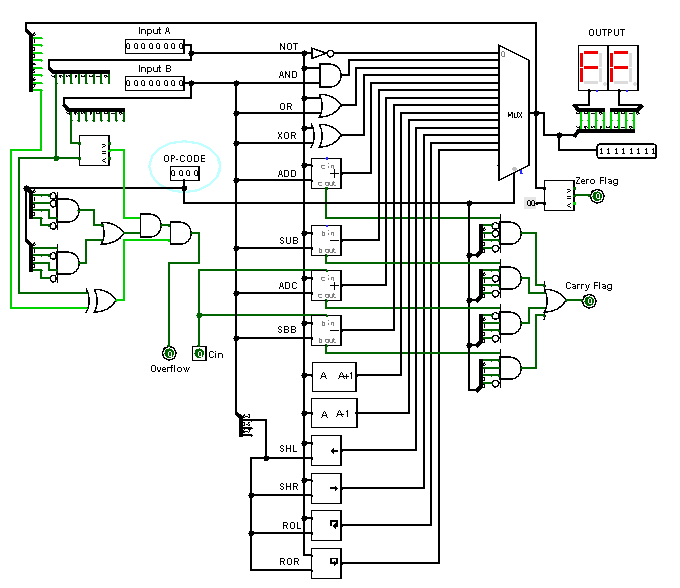
* A: es una entrada, que equivale al operando A.
* B: es una entrada, que equivale al operando B.
* Cin: en las operaciones donde se utiliza acarreo, el Cin se activa para que la operación la tome en cuenta.
* Función: es la operación que va a realizar (suma, resta, multiplicación, etc.) que esta dada por un código de operación (OP code) que indica que operación se va a realizar.
* Cout: indica que hay acarreo y desbordamiento en enteros si signo.
* Zero: indica si todas las líneas de resultados tienen valor de cero.
* Ov: indica si hay un desbordamiento en las funciones de suma y resta. Para los enteros sin signo, el indicador de *overflow* no es necesario.
* R: es el resultado de los dos operandos y de la función elegida.

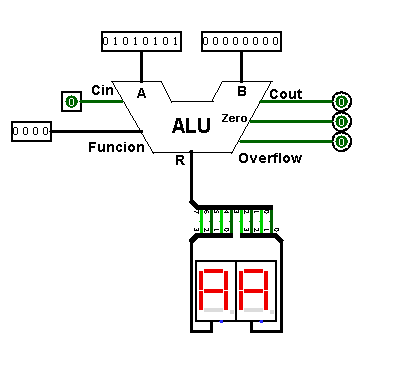
**Funciones de un ALU**

* **AND:** Esta compuerta es representada por una multiplicación en algebra booleana. Indica que es necesario que en todas sus entradas se tenga un estado binario 1 para que la salida otorgue un 1 binario.
* **OR:** Esta compuerta es representada por una suma en algebra booleana. Esta compuerta permite que con cualquiera de sus entradas que este en estado binario 1, su salida pasara a un estado 1 también.
* **XOR:** llamada OR exclusiva, esta actúa como una suma binaria de un digito cada uno y el resultado de la suma seria la salida. Otra manera de verlo es que con valores de entrada igual el estado de salida es 0 y con valores de entrada diferente, la salida será 1.
* **NOT:** En este caso esta compuerta solo tiene una entrada y una salida y esta actúa como un inversor. Para esta situación en la entrada se colocará un 1 y en la salida otorgara un 0 y en el caso contrario esta recibirá un 0 y mostrara un 1.
* **ADD:** realiza una operación de suma entre los dos operandos A y B.
* **ADC:** realiza una operación de suma con acarreo
* **SUB:** realiza una operación de resta entre los operandos A y B
* **SBB:** realiza una operación de resta con préstamo entre los operandos A y B
* **INC:** realiza un incremento de 1 al resultado de la operación.
* **DEC:** realiza un decremento de 1 al resultado de la operación
* **SHL:** realiza un desplazamiento a la izquierda
* **SHR:** realiza un desplazamiento a la derecha
* **ROL:** realiza una rotación a la izquierda
* **ROR:** realiza una rotación a la derecha.

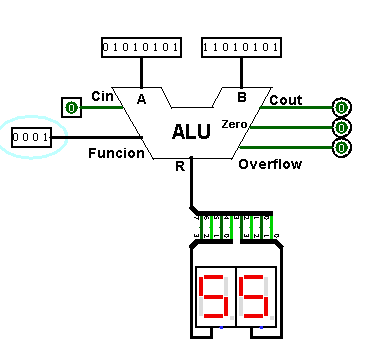
**Desarrollo**

Desarrollo de la ALU de manera detallada. Mas adelante se muestra todo esto simplificado en un solo circuito.

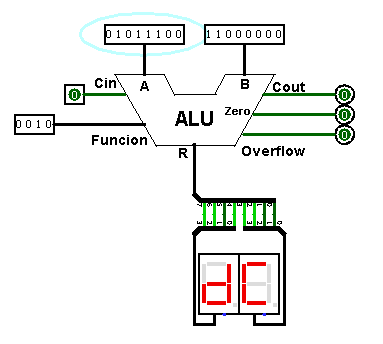


Aplicación de la ALU para todos los OP-code.

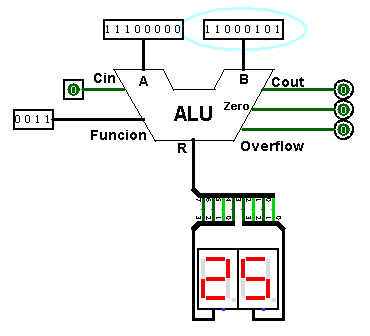
**Operación: NOT**



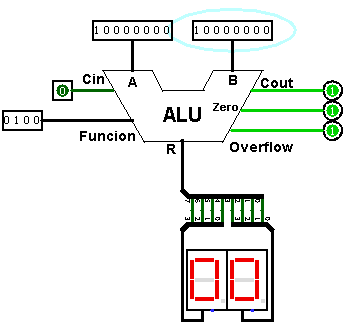
**Operación: AND**



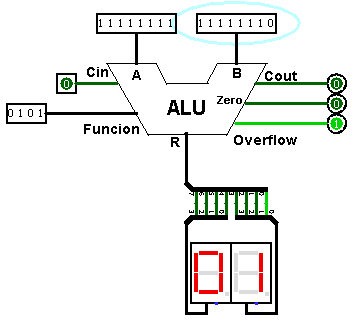
**Operación: OR**



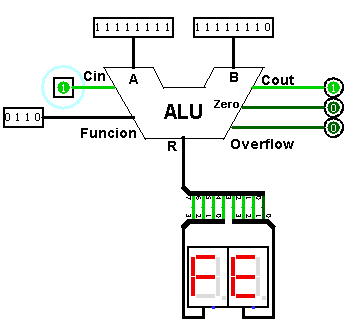
**Operación: XOR**



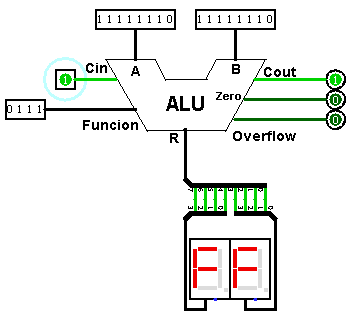
**Operación: ADD**



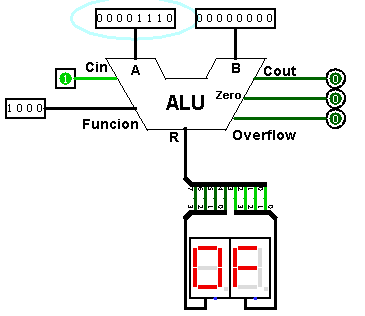
**Operación: SUB**



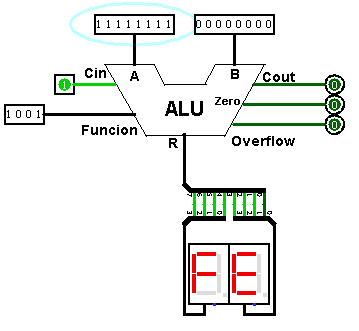
**Operación: ADC**



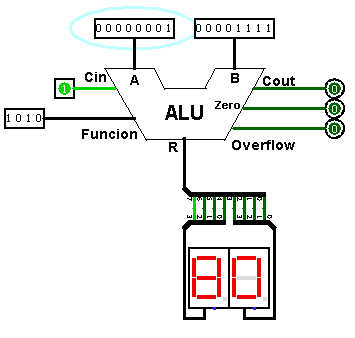
**Operación: SBB**



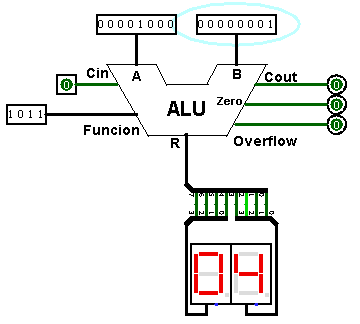
**Operación: INC**



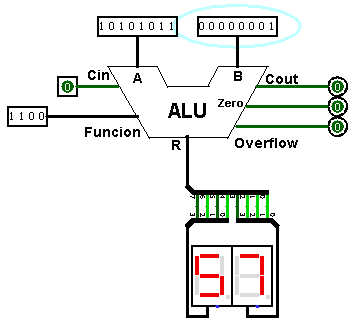
**Operación: DEC**



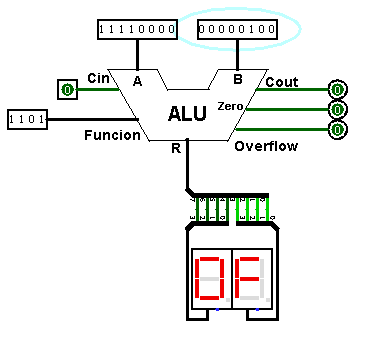
**Operación: SHL**



**Operación: SHR**



**Operación: ROL**



**Operación: ROR**

**Bibliografía**

Stallings W. (2016). *Computer organization and architecture: Designing for performance*. Pearson: New Jersey.

Barry B. (2009). *The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing*. Pearson: Ohio.