
ALU

Circuito digital ALU

Prof. Lic. Fernando PACHIOLI

¿Qué significa y qué es ALU?

ALU (Unidad aritmético lógica): dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

¿Cómo se usa?

1. La **UC** (unidad de control) suministra los operandos a la **ALU**
 2. La **UC** indica a la **ALU** la operación a llevar a cabo
 3. La **ALU** realiza la operación
 4. La **UC** toma el resultado
-

ALU - cont.

Componente fundamental en la arquitectura de una computadora que se encarga de realizar operaciones aritméticas y lógicas en los datos.

Es responsable de ejecutar diversas operaciones básicas en la unidad central de procesamiento (CPU). Algunas de las operaciones que puede realizar incluyen:

- **Operaciones aritméticas:** La ALU puede realizar operaciones como suma, resta, multiplicación y división en números binarios. Estas operaciones son esenciales para realizar cálculos matemáticos.
 - **Operaciones lógicas:** La ALU puede llevar a cabo operaciones lógicas, como AND, OR, XOR y NOT, en los bits de datos. Estas operaciones son utilizadas para evaluar condiciones, realizar comparaciones y tomar decisiones en función de los resultados.
 - **Desplazamientos y rotaciones:** La ALU puede realizar desplazamientos de bits hacia la izquierda o hacia la derecha, así como rotaciones de bits en una palabra de datos. Estas operaciones son útiles para desplazar y manipular datos en un registro.
-

ALU - cont.

La **ALU** trabaja en conjunto con otros componentes de la CPU, como los registros y los buses de datos, para realizar las operaciones necesarias. Recibe los datos de entrada, realiza las operaciones solicitadas y produce los resultados en el formato requerido.

Cabe destacar que el diseño y las características exactas de una **ALU** pueden variar dependiendo de la arquitectura de la computadora. Algunas **ALU** pueden ser más complejas y admitir operaciones adicionales, como operaciones de punto flotante en sistemas que manejan números decimales con precisión decimal.

Resumiendo

La **ALU** es una parte esencial de la CPU de una computadora y se encarga de realizar operaciones aritméticas y lógicas en los datos para ejecutar programas y llevar a cabo diversas tareas computacionales.



¿Para qué sirve?

Funciones y usos principales de la **ALU**:

- **Realizar operaciones aritméticas:** La **ALU** permite realizar operaciones matemáticas básicas, como suma, resta, multiplicación y división, en datos numéricos. Estas operaciones son utilizadas en una amplia gama de aplicaciones, desde cálculos simples hasta operaciones más complejas en áreas como la ciencia, la ingeniería y las finanzas.
 - **Ejecutar operaciones lógicas:** La **ALU** también es responsable de ejecutar operaciones lógicas, como **AND**, **OR**, **XOR** y **NOT**, en los datos binarios. Estas operaciones son fundamentales para evaluar condiciones, realizar comparaciones, tomar decisiones y controlar el flujo de ejecución en los programas.
-

¿Para qué sirve? - cont.

- **Manejar operaciones de desplazamiento y rotación:** La **ALU** puede llevar a cabo operaciones de desplazamiento y rotación de bits en palabras de datos. Estas operaciones son utilizadas para desplazar datos hacia la izquierda o hacia la derecha, rotar los bits en una palabra de datos y realizar manipulaciones específicas de bits.
 - **Soportar operaciones de punto flotante:** En algunas arquitecturas de CPU, la **ALU** puede tener la capacidad de realizar operaciones de punto flotante, lo cual es esencial para realizar cálculos con números decimales de precisión decimal.
 - **Procesamiento de datos en aplicaciones de inteligencia artificial y aprendizaje automático:** Con el crecimiento de la inteligencia artificial y el aprendizaje automático, la **ALU** juega un papel crucial en la realización de operaciones matemáticas complejas necesarias para el entrenamiento y la inferencia de modelos de machine learning.
-

Resumiendo

La ALU es responsable de llevar a cabo operaciones aritméticas, lógicas y de manipulación de bits en los datos. Es un componente esencial de la CPU que permite realizar cálculos, tomar decisiones y ejecutar programas en una computadora. Sin la ALU, la ejecución de tareas computacionales sería prácticamente imposible.



¿Cómo está construida?

La construcción exacta de una **ALU** puede variar dependiendo de la arquitectura de la CPU y el diseño específico de la ALU. Sin embargo, en general, la **ALU** está compuesta por diferentes componentes que trabajan en conjunto para realizar operaciones aritméticas y lógicas. A continuación, te menciono algunos de los componentes comunes que se encuentran en una **ALU**:

- **Unidad de suma:** La unidad de suma es responsable de llevar a cabo operaciones de suma en los datos. Puede incluir circuitos lógicos como sumadores completos o sumadores rápidos para realizar las sumas binarias.
 - **Unidad de resta:** La unidad de resta se encarga de realizar operaciones de resta. En algunos diseños, la unidad de suma también puede ser utilizada para realizar la operación de resta mediante la inversión de los bits y la adición de uno.
 - **Unidad de multiplicación y división:** En algunas **ALU** más complejas, puede haber unidades dedicadas para llevar a cabo operaciones de multiplicación y división. Estas unidades están diseñadas para ejecutar algoritmos específicos de multiplicación y división.
-

¿Cómo está construida? - cont.

- **Unidad de lógica:** La unidad lógica es responsable de realizar operaciones lógicas, como **AND**, **OR**, **XOR** y **NOT**. Utiliza circuitos lógicos para combinar y manipular los bits de los datos según las operaciones lógicas requeridas.
 - **Unidad de desplazamiento y rotación:** Esta unidad se utiliza para realizar operaciones de desplazamiento de bits hacia la izquierda o hacia la derecha, así como rotaciones de bits en una palabra de datos. Puede incluir registros y circuitos lógicos para realizar estas operaciones.
 - **Unidad de control:** La unidad de control coordina las operaciones de la ALU y se encarga de enviar las señales adecuadas a los diferentes componentes para realizar las operaciones requeridas. Controla el flujo de datos y las operaciones que se deben llevar a cabo.
-

¿Cómo está construida?

Además de estos componentes, una **ALU** también puede incluir registros para almacenar temporalmente los datos y los resultados de las operaciones, así como buses de datos para transmitir los datos entre los diferentes componentes.

Es importante tener en cuenta que la construcción exacta y los componentes específicos de una **ALU** pueden variar entre diferentes arquitecturas y diseños de CPU. Cada fabricante y arquitectura de CPU puede tener su propia implementación de la **ALU**, optimizada para el rendimiento y las características específicas del procesador.

¿Cómo se construye el “código” para operar y funcionar?

El código para operar o funcionar en una **ALU** (Arithmetic Logic Unit) se construye utilizando instrucciones de lenguaje de programación o código de máquina específico para la arquitectura de la **CPU** en la que se está trabajando. El código se escribe siguiendo un conjunto de reglas y sintaxis definidas por el lenguaje de programación o el conjunto de instrucciones de la **CPU**.

El proceso exacto de construir y ejecutar el código puede variar dependiendo del lenguaje de programación y la plataforma utilizada. Además, el código puede incluir otras instrucciones y operaciones adicionales que no están directamente relacionadas con la **ALU**, pero que son necesarias para el funcionamiento general del programa.

¿Cómo se construye el “código” para operar y funcionar?

Descripción general de cómo se construye el código para operar o funcionar en una **ALU**:

- **Selección de lenguaje de programación:** Primero, debes elegir un lenguaje de programación compatible con la arquitectura de la **CPU** en la que se encuentra la **ALU**. Los lenguajes de programación comunes incluyen C, C + +, Java, Python, entre otros.
 - **Escritura de código:** Utilizando el lenguaje de programación seleccionado, puedes escribir el código que describe las operaciones que deseas realizar en la **ALU**. Esto puede implicar la definición de variables, la especificación de las operaciones aritméticas o lógicas que deseas realizar, y la manipulación de los datos en la **ALU**.
 - **Compilación o interpretación:** Una vez que hayas escrito el código, debes compilarlo o interpretarlo en un formato ejecutable que la **CPU** pueda entender. Esto generalmente implica el uso de un compilador o un intérprete específico para el lenguaje de programación utilizado. El compilador traduce el código fuente a un código de máquina específico para la arquitectura de la **CPU**, mientras que el intérprete ejecuta el código línea por línea.
 - **Ejecución del código:** Una vez que el código se ha compilado o interpretado, puedes ejecutarlo en la **CPU**. El código contiene instrucciones que indican a la **CPU** cómo interactuar con la **ALU** y realizar las operaciones deseadas. Estas instrucciones se ejecutan secuencialmente, y la **CPU** utiliza la **ALU** para realizar las operaciones aritméticas o lógicas correspondientes.
-

Resumiendo

El código para operar o funcionar en una **ALU** se construye escribiendo instrucciones en un lenguaje de programación específico, que luego se compila o interpreta en un formato ejecutable. Estas instrucciones indican a la **CPU** cómo interactuar con la **ALU** y realizar las operaciones deseadas.



¿Cómo está compuesta una arquitectura ALU?

En términos generales, una **ALU** puede estar compuesta por los siguientes elementos:

- **Unidad de suma y resta:** La **ALU** incluye circuitos dedicados para realizar operaciones de suma y resta en números binarios. Estos circuitos pueden ser sumadores completos, sumadores rápidos o incluso sumadores paralelos más complejos, según los requisitos de rendimiento de la **ALU**.
 - **Unidad de lógica:** La unidad de lógica se encarga de realizar operaciones lógicas, como **AND**, **OR**, **XOR** y **NOT**, en los datos binarios. Esta unidad utiliza circuitos lógicos para combinar y manipular los bits de los datos según las operaciones lógicas requeridas.
 - **Unidad de multiplicación y división (opcional):** En algunas **ALU** más complejas, puede haber unidades dedicadas para realizar operaciones de multiplicación y división. Estas unidades están diseñadas para ejecutar algoritmos específicos de multiplicación y división y pueden incluir multiplicadores y divisores.
 - **Unidad de desplazamiento y rotación:** Esta unidad permite realizar operaciones de desplazamiento de bits hacia la izquierda o hacia la derecha, así como rotaciones de bits en una palabra de datos. Los circuitos en esta unidad pueden incluir registros y circuitos lógicos para realizar estas operaciones.
 - **Registros de entrada y salida:** La **ALU** puede tener registros dedicados para almacenar los operandos de entrada y los resultados de las operaciones. Estos registros permiten el almacenamiento temporal de datos durante las operaciones y la transferencia de datos dentro y fuera de la **ALU**.
 - **Unidad de control:** La unidad de control coordina las operaciones de la **ALU** y controla el flujo de datos y las operaciones que se deben realizar. Esta unidad se encarga de enviar las señales adecuadas a los diferentes componentes de la **ALU** para realizar las operaciones requeridas.
-

Resumiendo

La arquitectura de una **ALU** puede incluir una unidad de suma y resta, una unidad de lógica, una unidad de multiplicación y división (opcional), una unidad de desplazamiento y rotación, registros de entrada y salida, y una unidad de control para coordinar las operaciones y el flujo de datos en la **ALU**.



Circuito comparador de 1 bit

La arquitectura de un comparador de 1 bit es relativamente sencilla y se utiliza para comparar dos bits individuales y determinar si son iguales o diferentes. Aquí está la descripción básica de la arquitectura de un comparador de 1 bit:

- **Entradas:** El comparador de 1 bit tiene dos entradas, **A** y **B**, que representan los bits que se van a comparar. Estas entradas pueden ser señales digitales o bits almacenados en registros.
- **Circuito de comparación:** El circuito de comparación es el núcleo del comparador de 1 bit y se encarga de determinar si los dos bits de entrada son iguales o diferentes. El circuito puede utilizar compuertas lógicas como **XOR** para comparar los bits.
- **Salida:** La salida del comparador de 1 bit indica el resultado de la comparación. Puede ser una única línea que indica si los bits son iguales o diferentes, o puede ser una señal lógica que se utiliza en comparaciones posteriores en circuitos más complejos.

Es importante destacar que un comparador de 1 bit se utiliza como componente básico para construir comparadores de múltiples bits, que pueden comparar números de varios bits. Estos comparadores más grandes suelen seguir un diseño en cascada, donde se utilizan múltiples comparadores de 1 bit para comparar los bits individuales de los números.

Resumiendo

La arquitectura de un comparador de 1 bit incluye entradas para los dos bits que se van a comparar, un circuito de comparación que utiliza compuertas lógicas para determinar si los bits son iguales o diferentes, y una salida que indica el resultado de la comparación.

Este comparador básico se utiliza como componente en comparadores de varios bits para realizar comparaciones más complejas de números binarios.



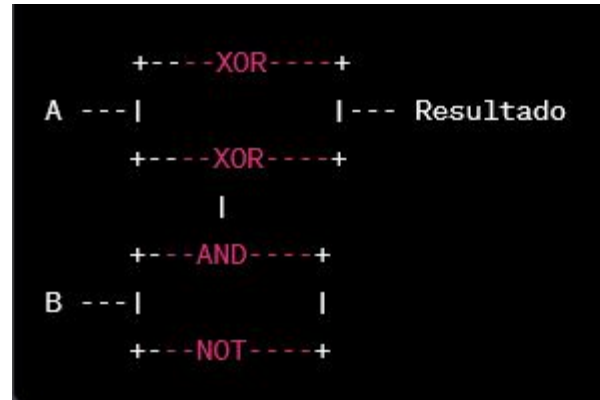
Comparador de 1 bit utilizando compuertas

Supongamos que queremos comparar los bits A y B:

```
A = 1  
B = 0
```

Comparador de 1 bit utilizando compuertas - cont.

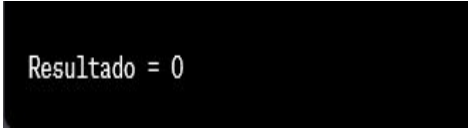
Utilizando compuertas lógicas, el circuito del comparador de 1 bit se vería así:



Comparador de 1 bit utilizando compuertas - cont.

En este caso, el **XOR** (OR exclusiva) se utiliza para comparar los bits **A** y **B**. Si los bits son diferentes, la salida del **XOR** será **1**; si son iguales, la salida será **0**. Luego, la salida del **XOR** se conecta a una compuerta **AND** junto con el bit **B** original. Esto es necesario para asegurar que tanto el bit **A** como el bit **B** sean iguales para que la salida del comparador sea 1.

En el ejemplo dado, como **A** es **1** y **B** es **0**, el resultado del comparador sería:



Resultado = 0

Comparador de 1 bit utilizando compuertas - cont.

Esto indica que los bits **A** y **B** son diferentes. Si **A** y **B** fueran iguales (por ejemplo, **A** = **1** y **B** = **1**), el resultado del comparador sería **1**, indicando que los bits son iguales.

Este es solo un ejemplo básico de un comparador de 1 bit utilizando compuertas lógicas. Los comparadores más grandes utilizan una estructura en cascada para comparar varios bits de números binarios.

Arquitectura de computadoras

sumador de 1 bit

La arquitectura de un sumador de 1 bit es uno de los componentes básicos utilizados en circuitos aritméticos y en la construcción de sumadores más grandes. Un sumador de 1 bit permite sumar dos bits de entrada (**A** y **B**) y producir una suma (**S**) y un acarreo (**Cout**) como resultado. Aquí tienes una descripción básica de su arquitectura:

1. **Entradas:** El sumador de 1 bit tiene dos entradas, **A** y **B**, que representan los bits que se van a sumar. Estas entradas pueden ser señales digitales o bits almacenados en registros.
 2. **Compuertas lógicas:** El sumador de 1 bit utiliza compuertas lógicas, como compuertas **XOR** (**OR exclusivas**) y compuertas **AND**, para realizar la suma.
 - La compuerta **XOR** se utiliza para sumar los bits **A** y **B**. La salida de la compuerta **XOR** se denomina "suma" (**S**) y representa el bit de salida de la suma de los dos bits de entrada.
 - La compuerta **AND** se utiliza para generar el acarreo (**Cout**) de la suma. La salida de la compuerta **AND** representa el bit de acarreo de la suma y se obtiene al realizar una operación **AND** entre los bits **A** y **B** de entrada.
 3. **Salidas:** El sumador de 1 bit produce dos salidas: la suma (**S**) y el acarreo (**Cout**). La salida de la suma representa el resultado de la adición de los dos bits de entrada, mientras que la salida de acarreo indica si hay un acarreo generado durante la suma. Estas salidas pueden ser utilizadas como entradas en sumadores de mayor tamaño.
-

Resumiendo

La arquitectura de un sumador de 1 bit incluye entradas para los dos bits de entrada (**A** y **B**), compuertas lógicas **XOR** y **AND** para realizar la suma y generar el acarreo, y salidas para la suma (**S**) y el acarreo (**Cout**).

Los sumadores de 1 bit son componentes fundamentales en circuitos aritméticos más complejos, como sumadores de varios bits o **ALU** (Arithmetic Logic Unit), que permiten realizar operaciones aritméticas más complejas en computadoras y procesadores.



Ejemplo de un sumador de 1 bit utilizando compuertas lógicas

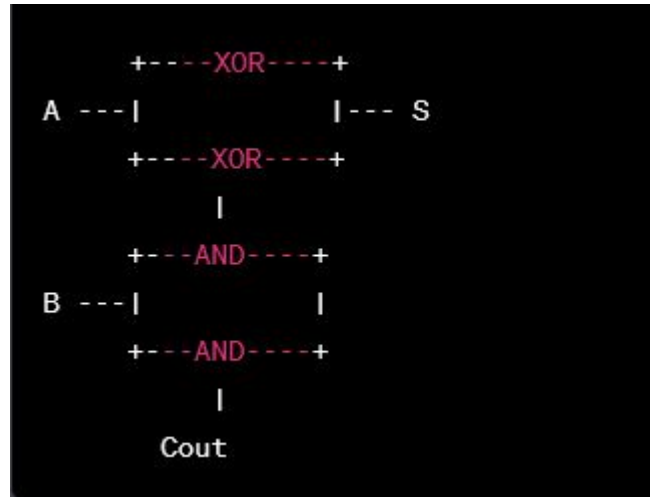
Supongamos que queremos sumar los bits A y B:

A = 1

B = 0

Ejemplo de un sumador de 1 bit utilizando compuertas lógicas - cont.

Utilizando compuertas lógicas, el circuito del sumador de 1 bit se vería así:



Ejemplo de un sumador de 1 bit utilizando compuertas lógicas - cont.

En este caso, el XOR se utiliza para sumar los bits A y B. Si los bits son diferentes, la salida del XOR será 1; si son iguales, la salida será 0. La salida del XOR se conecta a una compuerta AND junto con los bits A y B originales. Esto se hace para generar el acarreo (Cout) de la suma. La compuerta AND produce una salida de 1 si ambos bits A y B son 1.

Ejemplo de un sumador de 1 bit utilizando compuertas lógicas - cont.

En el ejemplo dado, como **A** es **1** y **B** es **0**, el resultado del sumador de 1 bit sería:

```
S = 1  
Cout = 0
```

Esto indica que la suma de los bits **A** y **B** es **1**, sin acarreo. Si **A** y **B** fueran **1**, el resultado sería:

```
S = 0  
Cout = 1
```

Ejemplo de un sumador de 1 bit utilizando compuertas lógicas - cont.

Esto indica que la suma de los bits **A** y **B** es **0**, con un acarreo de 1.

Este es solo un ejemplo básico de un sumador de 1 bit utilizando compuertas lógicas.

Los sumadores más grandes utilizan una estructura en cascada para sumar varios bits de números binarios.

Arquitectura de computadoras buses de un microprocesador

La arquitectura de un microprocesador incluye buses, que son vías de comunicación utilizadas para transferir datos y señales entre diferentes componentes del sistema. A continuación, te explicaré brevemente los tipos de buses comunes en un microprocesador y la máxima RAM direccionable.

- **Bus de datos** (Data Bus): Es un camino bidireccional por el cual se transmiten los datos entre el microprocesador y los demás componentes del sistema. El tamaño del bus de datos determina la cantidad de bits que se pueden transferir simultáneamente. Por ejemplo, un bus de datos de 32 bits puede transferir 32 bits de datos en cada operación.
 - **Bus de direcciones** (Address Bus): Es un camino unidireccional utilizado para especificar la ubicación de memoria o los registros que se están accediendo. El tamaño del bus de direcciones determina la capacidad de direccionamiento del microprocesador, es decir, cuánta memoria o registros puede acceder. Por ejemplo, un bus de direcciones de 16 bits puede direccionar hasta 64 KB de memoria ($2^{16} = 65,536$ bytes).
 - **Bus de control** (Control Bus): Es un conjunto de señales unidireccionales utilizadas para controlar y coordinar las operaciones entre los componentes del sistema. Estas señales pueden incluir señales de lectura/escritura de memoria, señales de control de temporización, señales de habilitación de dispositivos, entre otras.
-

Máxima ram direccionable

En cuanto a la máxima RAM direccionable, está determinada por el tamaño del **bus de direcciones**. **La capacidad máxima de memoria direccionable se calcula elevando 2 a la potencia del número de bits del bus de direcciones**. Por ejemplo, si un microprocesador tiene un bus de direcciones de 32 bits, la capacidad máxima de memoria direccionable sería de 2^{32} bytes, es decir, 4 GB (4,294,967,296 bytes). Sin embargo, es importante tener en cuenta que el tamaño máximo de la memoria direccionable puede estar limitado por otros factores, como la arquitectura del microprocesador o el sistema operativo utilizado.

Es fundamental considerar estas características al diseñar o trabajar con un microprocesador, ya que determinan la capacidad de transferencia de datos y la cantidad de memoria accesible.

—

—

continuará...

—