

# Organización y Arquitectura de Computadoras 2022-1

## Práctica 3: Unidad Aritmético Lógica

Profesor: José de Jesús Galaviz Casas \*

### 1. Objetivos

#### Generales:

- El alumno practicará el proceso de diseño de circuitos combinacionales.

#### Particulares:

Al finalizar la práctica el alumno estará familiarizado con:

- El diseño modular de los circuitos combinacionales.
- Los componentes básicos de una unidad aritmético lógica, así como el diseño de la misma.

### 2. Requisitos

#### ■ Conocimientos previos:

- Funciones de conmutación.
- Minimización de funciones de conmutación por medio de manipulación algebraica y mapas de Karnaugh.
- Los componentes básicos del diseño de circuitos combinacionales: transistores y compuertas **AND**, **OR** y **NOT**.
- Las funciones de una unidad aritmético lógica.

Se pueden consultar los temas en [Mano] y [Patterson].

- **Tiempo de realización sugerido:**  
5 horas.

---

\*Diseñada por Roberto Monroy Argumedo

- **Número de colaboradores:**

2

- **Software a utilizar:**

- *Java Runtime Environment* versión 5 o superior.
- El paquete *Logisim* [Logisim].

### 3. Planteamiento

Un componente elemental de un procesador es la unidad aritmético lógica o ALU (siglas en inglés de *Arithmetic Logic Unit*). Se diseñará y simulará una unidad aritmético lógica de 8 bits, que realice las siguientes operaciones:

- Las operaciones aritméticas de adición y sustracción.
- Las operaciones lógicas bit a bit de disyunción, conjunción y disyunción negada.
- Comparar si las entradas son iguales.
- Detección de desbordamiento aritmético.

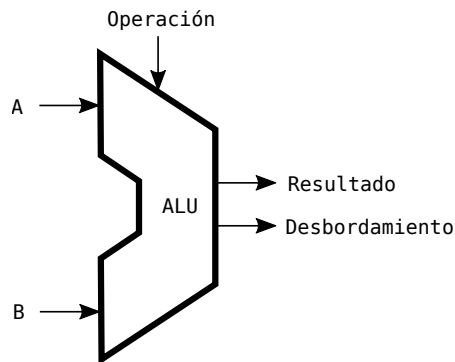


Figura 1: Unidad aritmético lógica.

### 4. Desarrollo

Para crear una ALU de 8 bits aprovecharemos la capacidad del diseño modular de los circuitos combinacionales y comenzaremos diseñando una ALU de 1 bit, posteriormente integraremos 8 ALUs de 1 bit para obtener una de 8 bits y finalmente agregaremos otros circuitos para obtener todas las funciones requeridas.

### 4.1. ALU de 1 bit

Nuestra ALU de 1 bit contará con las entradas  $A$ ,  $B$  y  $Op$  para los operandos y el código de operación respectivamente y una salida  $S$  para el resultado de la operación. Los primeros elementos que agregaremos serán para las operaciones **AND**, **OR**, y **NOT** ya que no necesitamos nada más que las compuertas lógicas, las entradas de éstas serán directamente las entradas  $A$  y  $B$  (En el caso del **NOT**, únicamente  $A$ ). Además incluiremos un sumador completo o *full adder*, agregando a la ALU una nueva entrada para el acarreo de entrada  $C_i$  y una salida para el acarreo de salida  $C_{i+1}$ . Basta con agregar un multiplexor para seleccionar la operación de salida, las entradas serán las salidas de las operaciones, el selector será  $Op$  y la salida  $S$ .

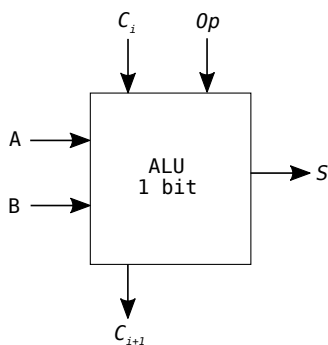


Figura 2: ALU de 1 bit con la capacidad de realizar las operaciones **AND**, **OR** y adición.

### 4.2. ALU de 8 bits

Obtenemos la ALU de 8 bits conectando ocho ALUs de 1 bit de la siguiente forma: la salida  $C_{i+1}$  de la ALU  $n$  se conecta a la entrada  $C_i$  de la ALU  $n + 1$ , como se muestra en la figura 3. Así cada  $A_n$ ,  $B_n$  y  $S_n$  corresponderán al  $n$ -ésimo bit del operando  $A$ , el operando  $B$  y la salida *Resultado*, comenzando con el bit menos significativo. La operación será la misma para cada una de las ALUs por que lo conectaremos directamente la entrada *Operacion* de la ALU de 8 bits a cada una de las ALUs de 1 bit. Queda sin conexión la entrada  $C_i$  de la primer ALU, la del bit menos significativo y la salida  $C_{i+1}$  de la última, la del bit más significativo; la primera nos será de utilidad más adelante en el desarrollo de la sustracción y la segunda nos sirve para detectar el desbordamiento aritmético, la cual sólo se deberá activar cuando se realiza la adición.

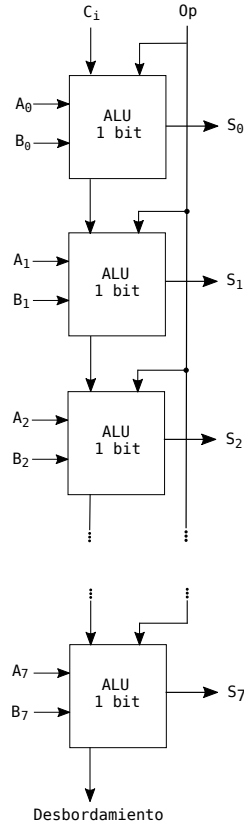


Figura 3: ALU de 8 bits a partir de ocho alus de 1 bit.

### 4.3. Sustracción

Primero adoptaremos la siguiente convención: para representar los inversos aditivos en nuestra ALU, usaremos la representación de complemento a dos. Entonces en el diseño del circuito que realizará la sustracción, podemos aprovechar los circuitos que ya tenemos, ésto es, para obtener  $A - B$  sumaremos al minuendo  $A$  el inverso aditivo del sustraendo  $B$ . Gracias a la convención adoptada, para obtener el inverso aditivo de un número, debemos invertir cada bit de la entrada  $B$  y sumarle 1, por lo que para obtener la sustracción, la ALU debe de llevar a cabo la operación  $A + \bar{B} + 1$ , en donde  $\bar{B}$  es la entrada  $B$  con todos sus bits negados.

#### 4.4. Comparaciones

De igual forma se puede aprovechar las operaciones ya construidas para llevar a cabo la comprobación de igualdad, menor que y mayor que, diseño que se deja como ejercicio.

Utilizaremos 0 como valor de falso y 1 como valor verdadero, es decir, las cadenas 00000000 y 00000001 respectivamente.

### 5. Entrada

El usuario podrá ajustar los bits de:

- Una entrada de 8 bits para el primer operando.
- Una entrada de 8 bits para el segundo operando.
- Una entrada de 3 bits para seleccionar la operación de acuerdo a los códigos especificados en la tabla 1.

Código operaciónn	Operación
000	AND
001	OR
010	NOT
011	Adicion
100	Menor que
101	Mayor que
110	Igualdad
111	Sustracción

Tabla 1: Códigos de operación.

### 6. Salida

- Una salida de 8 bits para el resultado de la operación.
- Una salida de 1 bit para indicar si ocurrió un desbordamiento aritmético.

### 7. Procedimiento

Deberás entregar un solo archivo de *Logisim* con las soluciones de los ejercicios y un documento con las respuesta a las preguntas planteadas.

Solamente puedes hacer uso de compuertas lógicas **AND**, **OR** y **NOT**, multiplexores, separadores y pines de entrada y salida. Recuerda etiquetar las entradas y salidas de cada uno de los subcircuitos.

## 8. Ejercicios

1. Desarrolla la ALU de 1 bit descrita en la sección 4.1.
2. Desarrolla la ALU de 8 bits descrita en la sección 4.2 utilizando el circuito del ejercicio anterior. Recuerda que la detección de desbordamiento aritmético sólo se debe activar cuando se solicita la operación adición.
3. Copia el subcircuito del ejercicio 2 en uno nuevo y extiéndelo agregando los componentes necesarios para llevar a cabo la sustracción descrita en la sección 4.3.
4. Extiende el circuito del ejercicio anterior y agrega los componentes necesarios para llevar a cabo la comprobación de igualdad, mayor y menor. Reutiliza las operaciones que ya tiene la ALU y considera: ¿qué relación hay entre el resultado de la operación  $A - B$  y el valor de verdad en las comparaciones  $A = B$ ,  $A < B$  y  $A > B$ ? Es importante notar que el código de operación ya no se pasa directamente al ALU del ejercicio 2 ya que esta recibe códigos de 2 bits para seleccionar la salida, por lo que es necesario agregar circuitos para detectar la operación y activar los componentes apropiados para la operación solicitada.
5. En un subcircuito coloca como un componente la ALU final del ejercicio anterior con pines de entradas y salida necesarios con el fin de llevar a cabo la simulación de la ALU de 8 bits.

## 9. Preguntas

1. ¿Qué operaciones aritméticas y lógicas son básicas para un procesador? Justifica tu respuesta.
2. El diseño utilizado para realizar la adición resulta ser ineficiente, ¿por qué? ¿Qué tipo de sumador resulta ser más eficiente?