Universidad Autónoma de Baja California

Facultad de Ciencias Quimicas e Ingenieria

ORGANIZACIÓN DE COMPUTADORAS Y LENGUAJE ENSAMBLADOR

Practica #3. Unidad Aritmética Lógica (ALU)

Integrantes

Caudillo Sanchez Diego

Osuna Angulo Omar Alonso

Docente

Evangelina Lara Camacho

Fecha de entrega

Viernes 14 de Septiembre de 2018

Práctica 3

Unidad aritmética y lógica

Objetivo

El alumno se familiarizará con la unidad aritmética y lógica de un sistema computacional.

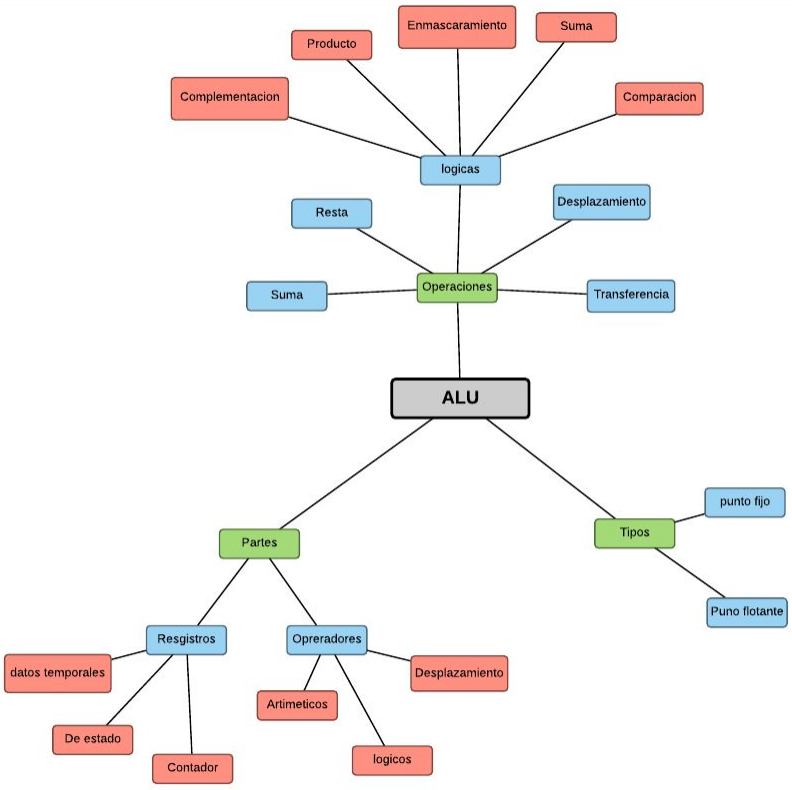
Equipo

Computadora personal con el software Logisim.

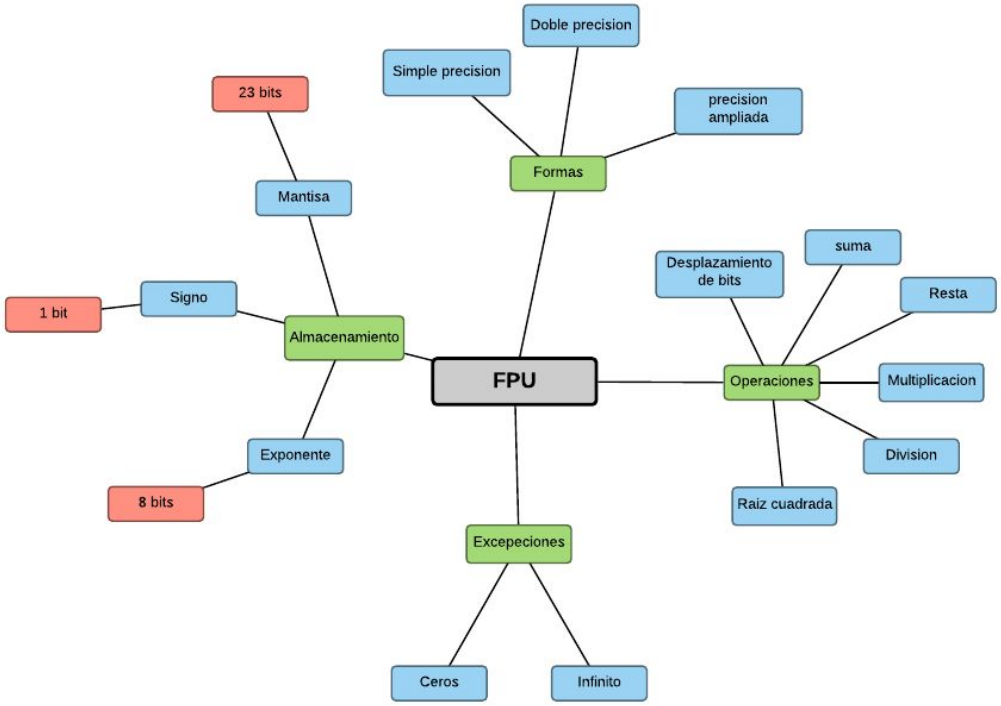
Teoría

Mapa mental sobre:

• **Unidad aritmética y lógica (ALU).**



• **Unidad de punto flotante (FPU).**



Responda las preguntas:

¿En qué situaciones se activa la bandera de acarreo (carry) en una ALU?

Una bandera de acarreo funciona con la ALU de un microprocesador. La manera en que la bandera de acarreo funciona se basa en cómo se realiza la suma y resta de números binarios.

Las reglas para que la bandera de acarreo se active en matemáticas de números enteros y binarios son dos:

1. **La bandera de acarreo se establece si la suma de dos números causa un acarreo de los bits más significativos (más a la izquierda) agregados.**

Un ejemplo de la bandera de carry activada podría ser en una **suma binaria** de dos números de 8 bits que se almacenará en un registro de 8 bits. Si el primer número es un **11111111** (255 decimal) y el segundo es un **00001001** (9 decimal), el resultado de la suma sería **100001000**

(264 decimal) y ocuparía un espacio de 9 bits para ser almacenado, sin embargo estamos limitados a guardarlo en un registro de 8 bits, por lo tanto se debe acortar el número a 8 bits, que sería el siguiente 00001000 (8 decimal). Al final se prende la bandera de carry, indicando que hay un acarreo debido a un resultado muy grande.

2. **La bandera de acarreo también se activa si la resta de dos números requiere un préstamo en los bits más significativos (más a la izquierda) restados.**

En el caso de una resta, cuando la secuencia 0000 es el minuendo y le restamos el valor binario 0001, el resultado es 1111, y la bandera de acarreo se activa, debido a que le estamos restando 0001 a 0000, el cual es un número menor. En decimal se vería como 0 - 1.

¿En qué situaciones se activa la bandera de sobreflujo (overflow) en una ALU?

La bandera se overflow solo es relevante para números con signo y no relevante para números sin signo. Un Overflow sólo puede pasar cuando sumamos dos números del mismo signo y obtenemos un signo diferente (en el MSB).

Las reglas para que la bandera de sobreflujo se active en matemáticas de números enteros y binarios son dos:

1. **Si la suma de dos números con los bits de signo desactivados o positivos (0) produce un número de resultado con el bit de signo activado (1) o negativo, se activa el indicador de OVERFLOW.**

Por ejemplo, de la siguiente suma binaria de dos números de 4 bits, donde su bit más significativo representa el signo del número, positivo en este caso (0). 0100 (4 decimal) + 0100 (4 decimal) = 1000 (indicador de sobreflujo está activado), el bit de signo cambia a 1 (negativo).

2. **Si la suma de dos números con los bits de signo en 1 (negativo) produce un número de resultado con el bit de signo desactivado (positivo), se activa el indicador de "desbordamiento".**

1000 + 1000 = 0000 (el indicador de sobreflujo está activado). De lo contrario, la bandera overflow no se encuentra activada.

Desarrollo

1. Diseñe y simule en Logisim una Unidad Aritmética y Lógica (ALU) que realice las funciones listadas en la Fig. 1. El tamaño de los operandos debe ser el indicado por el Instructor (32 bits).

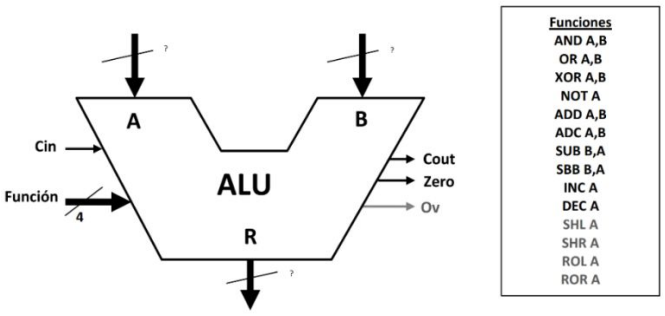
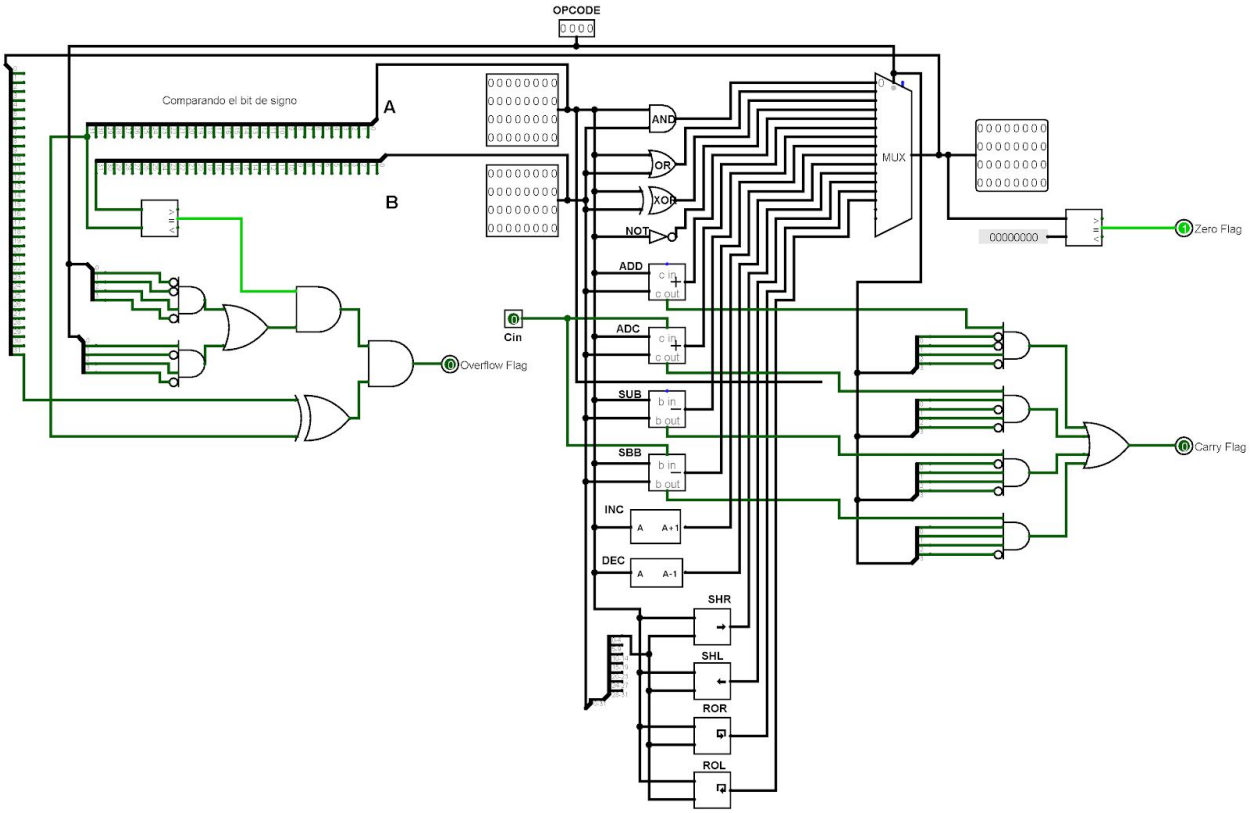


Figura 1. Unidad Aritmética y Lógica.



En esta implementación de la ALU, nuestro código de operación (**OPCODE**) nos permite acceder a una operación en específico de la ALU, por ejemplo, una suma con carry (**ADC**) tiene un opcode de 0101. El código de operación irá directamente al multiplexor para controlar sólo la salida de una operación a la vez. Cada operación tiene ligado un propio opcode, como se muestra a continuación:

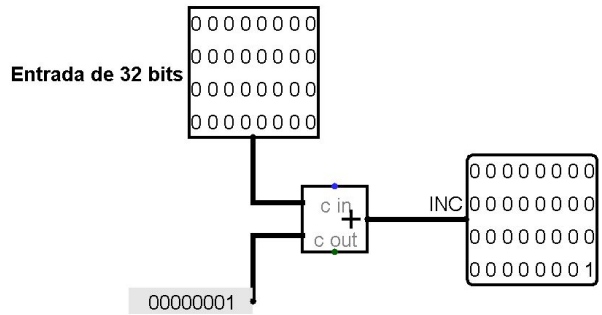
OPERACIÓN	CÓDIGO DE OPERACIÓN
AND **OR**	0000 0001
XOR	0010
NOT	0011
ADD	0100
ADC	0101
SUB	0110
SBB	0111
INC	1000
DEC	1001
SHR	1010
SHL	1011
ROR	1100
ROL	1101

Incluímos las banderas de acarreo, cero y de sobreflujo para que tuviera una funcionalidad aún mayor.

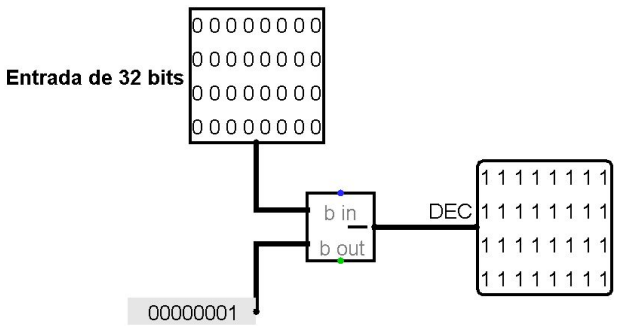
En las operaciones de SHR, SHL, ROR y ROL como únicamente se utiliza el operando A, ocupamos de otro valor para determinar el número de bits a desplazar, por lo tanto optamos por utilizar nuestro operando B, pasándolo por un splitter para obtener sólo los primeros 5 bits de la secuencia binaria (b0 - b4). Estos 5 bits serían los que nos ayudarían a determinar el valor de desplazamiento de nuestro número del operando A

En la parte de la función de incremento (**INC**) y decremento (**DEC**), creamos unos circuitos los cuales están compuestos de un sumador (ya creado por logisim) el cual tiene una entrada constante (un 1 binario), junto con la entrada que le demos. A continuación se muestra cómo están formados los dos circuitos:

Circuito responsable de la operación INCREMENT (INC)



Circuito responsable de la operación DECREMENT (DEC)



2. Describa paso a paso en el reporte el procedimiento de ejecución de las operaciones indicadas por el instructor.

- **SBB (Resta con préstamo)**

La resta con préstamo binaria es muy parecido a la resta decimal, lo que difiera de ambas es que en decimal cuando el minuendo es menor que el sustraendo se realiza un préstamo de 10 y en binario este préstamo es de 2.

Al igual que la resta en decimal, la resta binaria se inicializa desde los bits menos significativos. Entonces para poder realizar una resta con préstamo se necesita dos elementos: **minuendo** y **sustraendo**, que son los números que se van a restar.

Tomando como ejemplo lo anterior, suponemos que A=1101 y B=0110.

Ejemplo: A-B.

```
   11
  011
_ 1101 ← A

  0110 ← B
----------
  0111 ← Resultado
```

Bit 1: En el bit uno nos damos cuenta que no hay ningún problema con la resta ya que cumple con la condición, donde el minuendo es mayor o igual al sustraendo. Como resultado tenemos un **1.**

Bit 2: En el segundo bit se necesita un préstamo del **bit 3** ya que el sustraendo es mayor que el minuendo, por lo que ahora en el **bit 2** tenemos 11, al ser el **bit 3** el **doble de valor** que el **bit 2** y al restarle **1** del sustraendo, se elimina uno de los 1's prestados del bit 3 y como resultado final en el **bit 2** tenemos un **1**.

Bit 3: En consecuencia de la operación anterior, el **bit 3** se ve afectado por el préstamo realizado, y el resultado en el es de **0**. Al hacer la operación con el resultado actual se presenta la condición que el sustraendo es mayor que el minuendo por lo que volvemos a realizar un

préstamo a bit más significativo siguiente que es el **bit 4**. Al igual que en el bit 2, pasa lo mismo con el bit 3, ahora tenemos 11, y al restarle el 1 del sustraendo se elimina un 1 y se almacena en el **bit 3** del resultado final un **1**.

Bit 4: Finalmente el **bit 4** se ha quedado en **0** y al restarle el 0 cero sustraendo tenemos en el bit 4 del resultado final un **0**.

- **ROL (Rotación a la izquierda)**

La rotación es muy parecida al desplazamiento de bits, sólo cambia la manera en la que se usa el bit más significativo para reemplazar al lugar en donde se encontraba el bit menos significativo. Por lo tanto no se pierde nunca el valor del MSB.

Una Rotación, al igual que un desplazamiento de bits, es una operación bit a bit o también conocida como *bitwise*. Esto significa que opera sobre números binarios a nivel de sus bits individuales.

Entonces para realizar una rotación a la izquierda se requiere de dos elementos: **el número a rotar**, y **la cantidad de bits a rotar**. Al realizar una rotación, el bit de la posición **n** es reemplazado por el bit de la posición **n-1** y al llegar al bit más significativo, este se recorre a la posición del bit menos significativo. Todo éste proceso se repite tantas veces como sea indicado por la cantidad de bits a rotar.

Suponiendo que tenemos el número 1001, y queremos rotarlo hacia la izquierda 2 bits, nuestro resultado sería 0110, debido a que al hacer una rotación a la izquierda, dejamos al bit más significativo fuera de la secuencia binaria (tomado como acarreo) y lo utilizamos para tomar el espacio del bit menos significativo, este proceso se va repitiendo las veces que se indiquen.

Número a rotar: 1001
Cantidad de bits a rotar: 2

- 1era rotación: 0011
- 2da rotación: 0110

Para que un número rotado vuelva a su estado inicial, habrá que rotarlo **N** bits, siendo **N** la extensión en bits del número a tratar.

Conclusiones y comentarios

Diego Caudillo Sanchez:

El uso de la ALU nos permitió realizar distintas operaciones, así como entender su funcionamiento y los pasos que realiza algunas de sus operaciones aritméticas como lo fue la resta, ya que esta última difiere un poco al tipo de resta que se utiliza en el decimal.

Otro punto importante que tiene la ALU son sun banderas, como la de *overflow* y *carry*. Las cuales nos permiten detectar errores a la hora que una operación se está llevando a cabo. La cual indica si el resultado de las operaciones superan el tamaño del tipo de dato en el que se almacenan las operaciones, activándose en este la *overflow flag*. Por otro en *carry flag* obtenemos un error aritmético esta bandera se activa para avisar al programador del error.

Omar Alonso Osuna Angulo:

La ALU es una unidad indispensable para la composición de un Microprocesador, sin ella no se podrían hacer operaciones aritméticas con números enteros. Hay ALU's de distintas capacidades, donde el código de operación nos dicta cuántas operaciones puede realizar, siendo las más importantes la Suma, Resta, AND, OR, XOR.

Cada que la ALU realiza alguna operación, ésta tiene a la salida un resultado de la operación (en función de sus operandos) y a su vez, un resultado del estado de la operación (Overflow, Carry, Zero, etc.) que nos indica cómo es el resultado.

Por lo visto, el acarreo es usado generalmente para manejar aritmética sin signo, mientras que el sobreflujo es usado para aritmética con signo (cuando usamos complemento a 2). El carry sólo nos es relevante cuando manejamos números sin signo. Cuando manejamos números sin signo, la salida de la bandera sobreflujo será irrelevante.

Cuando se activa la bandera de carry al hacer una suma es porque el resultado de la suma sobrepasa la cantidad de bits que están disponibles para guardar el resultado y como consecuencia, tenemos un 1 acarreado. Mientras que en una resta la bandera de carry se activa cuando se requiere tomar un préstamo debido a que el minuendo es más pequeño que el sustraendo.

Dificultades en el desarrollo

Diego Caudillo Sanchez:

La parte más complicada fue la implementación de las banderas de acarreo y sobreflujo, ya que en un principio tuvimos problemas, en ocasiones ambas se encendían cuando no tenían que hacerlo, por lo que optamos en utilizar el principio del decodificador para que solo funcionaran con las operaciones seleccionadas como sumar y restar.

Otro contratiempo fueron las operaciones de incremento y decremento. En un inicio lo confundimos con agregar un bit, pero esto no cumplia con la funcion real que era restar o sumar un bit. Finalmente se utilizó una herramienta que ofrece logisim para poder realizar esta acción.

Omar Alonso Osuna Angulo:

Una de las cosas que se nos dificultó fue determinar de qué forma podríamos representar la bandera de sobreflujo, que en algunas ocasiones se activa al mismo tiempo que la bandera de acarreo. Para ello utilizamos un comparador de bits para comparar el último bit de los dos operandos junto con el resultado de la operación. También verificamos que la bandera de sobreflujo y de acarreo estuviera activa sólo cuando el código de operación le correspondiera a las operaciones que nos interesan (suma y resta), para ello usamos una especie de decodificador de memoria.

Otro impedimento fue que no sabíamos cómo realizar un circuito que hiciera la operación de INC y de DEC, por lo tanto se tuvo que investigar más a fondo cómo hacer tal operación.

Referencias

- ➔ https://www.techopedia.com/definition/12419/carry-flag-c-flag
- ➔ http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt
- ➔ https://searchwindowsserver.techtarget.com/definition/floating-point-unit-FPU
- ➔ https://x-engineer.org/graduate-engineering/embedded-systems/microcontrollers/bit-shift-rotation-algorithms-scilab/