Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática
CI-0119 Proyecto Integrador de Lenguaje Ensamblador y Arquitectura
II ciclo de 2018

Integrantes del equipo:
Lester Cordero Murillo - B62110
Josué Choso Rojas - B62045
Jesús Mena Amador - B54291
Emmanuel Palma Ortiz - B45147

1. Implementación detallada de nuestra calculadora

Esta sección dará inicio a la información fundamental de la calculadora explicando de forma detallada el como será su arquitectura, sus instrucciones y el cómo funciona cada una de sus subpartes. A continuación se detallara de forma específica todos los componentes de logisim usados y posteriormente el porque decidimos usarlo de dicha manera.

1.1 Descripción del Circuito de entrada

1.1.1 Funcionamiento externo del circuito de entrada

Este circuito consiste de 12 entradas controladas por diferentes botones asociados a diferentes acciones:

Entradas/Teclas 0 − 9: Corresponden a las entradas numéricas. Estas internamente utilizan un codificador de prioridad que convierte su orden de entrada en el valor que representan y lo envían a 4 registros simples de 16 bits cada uno. Usualmente el bcd solo necesita 4 bits por dígito, sin embargo debido a que necesitaremos convertir todos los dígitos bcd a un solo gran número hexadecimal de bastante precisión (osea 16 bits), justificamos que se necesita que cada dígito sea de este mismo para que logisim nos deje hacer la conversión de forma mas natural.

Entrada/Tecla n: Activa una bandera que indica si el número ingresado será complementado a negativo o viceversa. En caso de que la bandera este encendida, se prenderá el LED rojo con la etiqueta “negativo” y enviara inmediatamente una interrupción que puede ser detectada por el ROM de instrucciones.

Entrada/Tecla C: Borra el número que el usuario estaba ingresando usando el pin de ”borrado asincrónico” incluido en los registros de logisim.

Entrada/Tecla AC: Igual que “C” pero además borra todos los números guardados en la pila usando un mismo bus.

Entrada/Tecla E: Corresponde a la tecla Enter, y consiste en copiar mediante la salida resultante del circuito de entrada principal, el número que el usuario ingreso (el que se encuentra en pantalla) en formato bcd, convertirlo a hexadecimal, y transcribirlo a el top de la pila. Siempre realiza un shift hacia abajo (dentro de la pila) al colocar el elemento en esta.

1.1.2 Funcionamiento interno del circuito de entrada

EL funcionamiento interno de este circuito de entrada basicamente consiste en convertir un número bcd a hex. A continuación detallaremos como se realiza esta conversión:

Existe 1 registro para cada dígito bcd que tenga la calculadora. Cuando el usuario ingresa un número, realiza un shift de todos los registros hacia la izquierda (en la pantalla) y luego inmediatamente copia el número correspondiente a la tecla presionada al registro menos significativo. Al terminar multiplica cada dígito por una potencia de 10, para obtener su equivalencia hexadecimal.

Las unidades las deja tal cual, las décimas por 10, las centésimas por 100, etc y luego suma los resultados de cada una de las multiplicaciones almacenándolo en un único registro que contendrá el número bcd correctamente convertido a hexadecimal.

1.2 Descripción del Circuito de salida

1.2.1 Funcionamiento externo del circuito de salida

De forma externa, posee dos entradas, 1 salida para el bit de signo y multiples salidas para cada dígito bcd. La primera entrada corresponde al "Refresh" y la segunda entrada corresponde un dato hexadecimal. Únicamente cuando refresh reciba un 1, es cuando el circuito de salida cargara el dato en su buffer interno. Esto permite que este circuito se pueda controlar desde instrucciones.

Además posee un mux exterior conectado al bus de datos, en donde este sirve para escoger desde donde se obtendrá el número que será presentado en la pantalla. Esto puede ser, del teclado numérico, de la pila o de algún resultado del ALU.

1.2.2 Funcionamiento interno del circuito de salida

Muy similar al de circuito de entrada pero realizando el proceso al revés, pues recibe un único número hexadecimal a cada uno de los dígitos bcd ylos coloca en registros buffer internos separados.

El algoritmo consiste en que para cada iteración, comprobar si cada registro tiene un número mayor que 4, y caso afirmativo, le suma 3 y realiza un shift; caso contrario, solo realiza el shift sin sumarle nada y vuelve a repetir otra iteración. El algoritmo termina al realizar 16 desplazamientos. El número resultante, tendrá múltiples salidas en donde cada una contendrá el dígito bcd. Existe un bit extra que corresponde al signo, en donde complementa el número a su negativo en caso de ser necesario.

1.3 Descripción del Sistema de instrucciones

La calculadora posee un decodifcador general, que escoje según los datos que provengan de la ROM, que instrucción ejecutar. Escoje solamente el conjunto de 3 o 4 bits mas significativos como instrucción y los 4 menos significativos seràn usados como parámetros para alterar la instrucción.

Por el momento todas nuestras instrucciones son de un parámetro, pero según vamos optimizando la calculadora, es posible que reduzcamos el número de instrucciones a favor de hacer que las instrucciones tengan mas parámetros para que se pueda hacer mas acciones con una sola instrucción variada en lugar de usar varias instrucciones similares. Las instrucciones serán guardadas en la memoria ROM.

1.3.1 Composición de las instrucciones

Esta tabla representa el formato que usan nuestras instrucciones. Si en el futuro decidimos agregar mas parámetros a las instrucciones, reemplazaremos "n" por el número que nos convenga. Actualmente es solo hasta el argumento 1.

Bit mas significativo	**Bits intermedios**	**Bit menos significativo**
1 dígito hex OPCODE: Operación	1 dígito hex ARG(1): Parámetro	1 dígito hex ARG(n): Parámetro n

1.3.2 Tabla de instrucciones

Corresponde al código exacto que se debe ingresar en la rom para ejecutar una función dada. A continuación se detalla:

Instrucción	**Descripción**
00	**Instrucción Nula (No realiza ningún procedimiento)**
1X	**Familia de instrucciones: MOV {source}**
10	Coloca en pantalla un número hexadecimal proveniente de lo que el usuario escribió del teclado.
11	Coloca en pantalla un número hexadecimal proveniente de la suma de los dos primeros números de los registros de la pila.
12	Coloca en pantalla un número hexadecimal proveniente de la resta de los dos primeros números de los registros de la pila.
13	Coloca en pantalla un número hexadecimal proveniente de la multiplicación de los dos primeros números de los registros de la pila.
14	Coloca en pantalla un número hexadecimal proveniente de la división de los dos primeros números de los registros de la pila. Coloca el residuo en la pila.
15	Coloca en pantalla un número hexadecimal que corresponde al factorial entre los dos primeros números de los registros de la pila.
16	Coloca en pantalla un número hexadecimal que corresponde a la potencia entre los dos primeros números de los registros de la pila. Eleva el primero al segundo.
2X	**Familia de instrucciones: JMP {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir".
3X	**Familia de instrucciones: NUM {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si el teclado numérico no ha emitido ninguna interrupción.
4X	**Familia de instrucciones: ENT {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla ENTER no ha emitido ninguna interrupción.
5X	**Familia de instrucciones: ALU (Sin Argumentos)** Realiza inmediatamente todos los cálculos aritméticos simultaneamente. Pero para extraerlos, se necesita de "MOV". Debido a que FACT Y POW duran varios ciclos, necesitan más banderas

Instrucción	Descripción
6X	**Familia de instrucciones: SUM {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla SUMA no ha emitido ninguna interrupción.
7X	**Familia de instrucciones: SUB {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla RESTA no ha emitido ninguna interrupción.
8X	**Familia de instrucciones: MUL {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla MULTIPLICACIÓN no ha emitido ninguna interrupción.
9X	**Familia de instrucciones: DIV {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla DIVISIÓN no ha emitido ninguna interrupción.
AX	**Familia de instrucciones: FACT {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla FACTORIAL no ha emitido ninguna interrupción.
BX	**Familia de instrucciones: POW {dir}** Realiza un salto de memoria del punto actual hacia la dirección "dir" solamente si la tecla POWER no ha emitido ninguna interrupción.
CX	**Familia de instrucciones: WAIT {flag}** Detiene la ejecución de todo el programa, hasta que se active alguna bandera especifíca.
C0	Espera hasta que se active la bandera FACTORIAL desde el ALU, que indica que ya termino de calcularse.
C1	Espera hasta que se active la bandera POWER desde el ALU, que indica que ya termino de calcularse.
DX	**Familia de instrucciones: STACK {push/shift}** Realiza entrada y salida de datos a la pila
D0	Realiza un push en la pila, obteniendo el dato desde el teclado numérico.
D1	Realiza un shift vertical en la pila, similar a un "pop".

DOCUMENTO INCOMPLETO WORK IN PROGRESS

DOCUMENTO INCOMPLETO WORK IN PROGRESS

DOCUMENTO INCOMPLETO WORK IN PROGRESS