Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Curso: CE-4301 Arquitectura de Computadores I
(Course: CE-4301 Computer Architecture I)

Documento de Descripción de la Microarquitectura
(First Project)

Realizado por:
Made by:
Fabián Astorga Cerdas 2014040808
Javier Sancho Marín 2014159997
Óscar Ulate Alpízar 201229559

Profesor:
(Professor)
Fabián Zamora Ramírez

Fecha: Cartago, 17 octubre, 2017
(Date: Cartago, October 17, 2017)

MICROARQUITECTURA JOF32

1. Overview

Los dos componentes principales de la arquitectura que se presentan en el documento son la ruta de datos, la cual presenta cada uno de los componentes y sus conexiones que permiten que se ejecuten las instrucciones y que procesan cada uno de los datos, y la ruta de control, la cual contiene todos los componentes que permiten que se generen señales de control que manipulan los componentes de la ruta de datos con el fin de que todo se procese adecuadamente.

Los componentes de control y algunos de la ruta de datos contienen la señal más importante de todas que es el *clock,* el procesador trabaja sobre un ciclo, estos ciclos son proporcionales a la velocidad del cambio del *clock,* la frecuencia con la que cambia el *clock* en esta arquitectura es de 2 GHz.

Debido a la organización de la arquitectura, se presenta cada una de las etapas por separado, este procesador está diseñado sobre una organización pipeline de 5 etapas:
Fetch, Decode, Execute, Memory y *Write Back.*

2. Etapa de búsqueda (*Fetch*)

En la figura 2.1, se muestra la organización de la etapa de búsqueda.

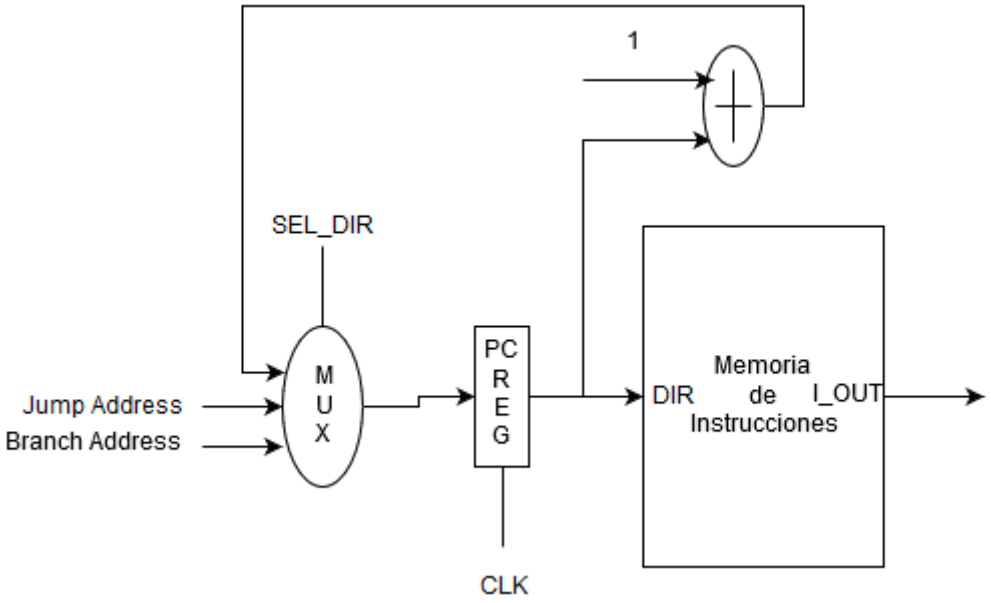

Figura 2.1. Etapa de búsqueda de la microarquitectura.

a. Ruta de datos

Dentro de las decisiones tomadas para la organización de la arquitectura se consideró crear la memoria de instrucciones de un tamaño de 1 KB lo que nos permite que el valor de PC (program counter) sea de un máximo de 1024 el cual se puede representar con 10 bits.

Al tener únicamente 10 bits para el PC se disminuye la cantidad de buses de datos necesarios y la potencia que consume el procesador.

La salida de la memoria de instrucciones si de 32 bits debido a que las instrucciones tienen ese largo, las cuales están descritas detalladamente en el documento de Descripción del ISA.

La ruta de datos de esta etapa está compuesta por:

- MUX: de 3 entradas de 10 bits cada una, este permite el paso del nuevo valor de la señal de PC que es el que indica la instrucción a ejecutarse en el siguiente ciclo de reloj.
- PC_REG: se encarga de guardar el valor del PC conforme transcurren los ciclos de reloj.
- Memoria de instrucciones: en esta se guardan las instrucciones generadas por el programador, estas instrucciones son pre procesadas adecuadamente por un compilador.
- Sumador: la única función es realizar una suma al PC, siendo el resultado PC+1, con el fin de aumentar este contador y continuar con la instrucción siguiente.

b. Control

En esta etapa no se requiere de muchas señales de control ya que únicamente maneja el PC, la señal de control utilizada es la de SEL_DIR. Esta se encarga de elegir cuál será el valor del PC dependiendo de la instrucción ejecutada. Si es una instrucción JUMP, el MUX elegirá la señal jump address, si es una instrucción BRANCH, el MUX elegirá la señal branch address, de lo contrario, eligira la señal proveniente del sumador.

3. Etapa de decodificación (*Decode*)

En la figura 3.1, se muestra la organización de la etapa de decodificación.

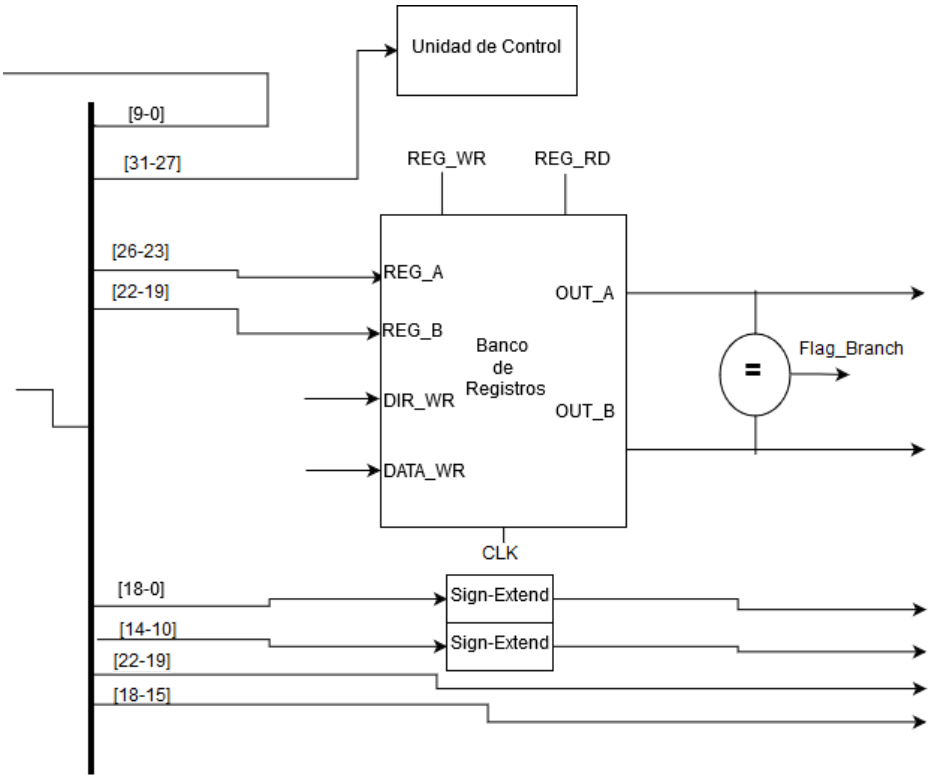

Figura 3.1. Etapa de decodificación de la microarquitectura.

a. Ruta de datos

La instrucción tomada de la etapa anterior se descompone en muchos buses de datos que se van ejecutando paralelamente aunque no se requiera de todos, esto con el fin de no tomar decisiones en el proceso y atrasar más la ejecución de la instrucción, en la siguiente etapa se realizará el manejo de estos datos tomados.

Las separación de la instrucción se detallada de la siguiente manera:

- [31-27]- OPCODE
- [26-23]- Dirección de registro RS
- [22-19]- Dirección de registro RT
- [18-15]- Dirección de registro RD
- [14-10]- Valor de SHAMT
- [18-0]- Valor del inmediato
- [9-0]- Valor de PC en caso de salto

Los componentes que conforman esta etapa casi todos son asíncronos, no requieren de la señal de *clock* a diferencia del banco de registros, ya que paralelamente estará leyendo y escribiendo en el banco, esto se logra por medio de la toma del cambio positivo del *clock* (de nivel bajo a nivel alto) como lectura y del cambio negativo del *clock* (de nivel alto a nivel bajo) como escritura.

Los componentes de la etapa son:

- Banco de registros: Este banco es de 16 registros de 32 bits cada uno. En caso de lectura se utiliza las entradas de REG_A, REG_B con ayuda de estos dos se obtienen los datos en OUT_A y OUT_B. Para la escritura se utiliza DIR_WR la cual indica en cual registro se escribirán los datos de DATA_WR, estas dos últimas señales vienen de la etapa de write back.
- Sign-Extend: Extiende el signo de la señal de entrada a una señal de salida de 32 bits
- Comparador: Permite conocer el resultado de la comparación que se necesita en las instrucciones branch y enviar esta señal a la unidad de control para su adecuada interpretación.
- Unidad de control: Maneja por medio de una máquina de estados todas las señales del procesador.

b. Control

Las señales de control necesarias son las de REG_WR y REG_RD, estas manejan el banco de registro, la señal REG_WR en caso de que se requiera escribir, o si no se desea escribir nada en caso de que la instrucción no lo requiera, la señal REG_RD para leer del banco en caso de que se requiera.

4. Etapa de ejecución (*Execute*)

En la figura 4.1, se muestra la organización de la etapa de ejecución.

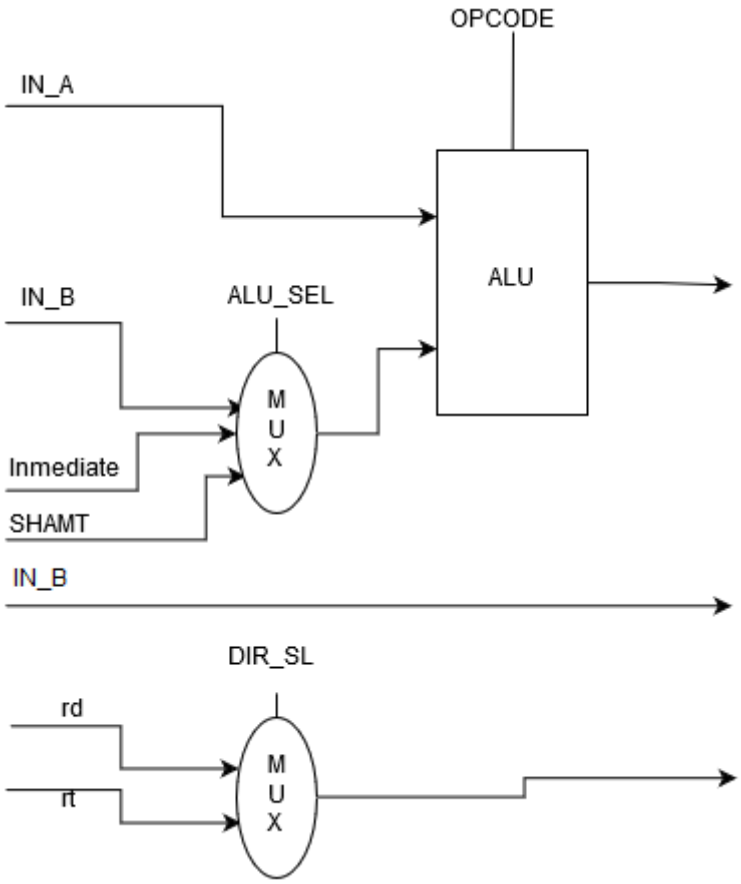

Figura 4.1. Etapa de ejecución de la microarquitectura.

a. Ruta de datos

Todos los datos en esta etapa se toman del registro ID/EXE, incluyendo el valor inmediato de las instrucciones tipo I. Este valor es necesario pasarlo por el registro ID/EXE para no perder la integridad de la instrucción.

El dato In_A pasa directamente a la entrada A de la ALU. El dato In_B, por el contrario, pasa primero por un MUX. En este MUX se decide por medio de control si debe pasar el dato In_B o el valor inmediato de las instrucciones I que viene del registro ID/EXE. Este MUX es controlado por la "nube de control" de la microarquitectura.

La ALU como tal, contiene dos entradas de datos de 32 bits, una entrada de control que contiene el OPCODE de la instrucción y una salida que lleva el dato resultado hasta el registro EXE/MEM. Todos las operaciones que efectúa la ALU se ejecutan sobre alguno de los dos entradas de datos. Cabe destacar que el dato In_B comparte un MUX con el dato

inmediato y con el SHAMT de las instrucciones SHIFT. En ese caso, al dato que se le va a aplicar el corrimiento debe ingresar por la entrada In_A.

Por último hay un MUX que permite el paso sólo del registro RD o RS para escribir datos en esas direcciones en la etapa de WB.

b. Control

Las señales de control utilizadas en la etapa de execute son las siguientes:

- OPCODE: La ALU utiliza esta señal para determinar cuál operación realizar
- ALU_SL: Selecciona cual de los datos se opera en la ALU.
- DIR_SL: Selecciona cual de las dos direcciones de registro será en la que se guardará el dato luego de finalizada la instrucción.

5. Etapa de memoria (*Memory*)

En la figura 5.1, se muestra la organización de la etapa de memoria.

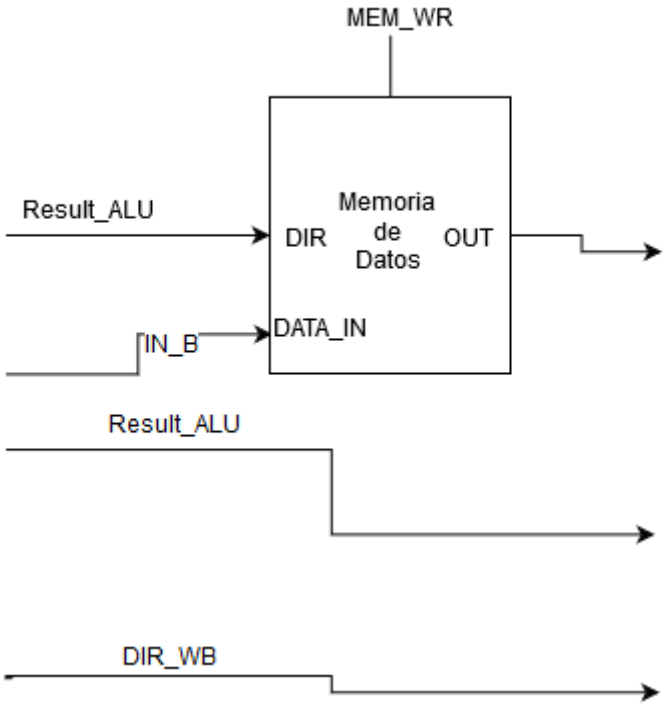

Figura 5.1. Etapa de memoria de la microarquitectura.

a. Ruta de datos

La memoria tiene como ingreso dos líneas de datos principales, una para la dirección de memoria donde se va a leer o escribir y otra línea que ingresa el dato que se va a escribir en caso de un Store Word. La memoria tiene solo una salida. Esta es una línea de 32 bits que lleva el dato obtenido de memoria en caso de que la operación sea un Load Word o un Load Byte.

Cuando la operación es un Load Byte, la memoria se encarga a lo interno de generar como salida sólo 8 bits. El registro de salida va a tener 32 bits de ancho, sin embargo, los bits [31,8] van a ser ceros en todas las ocasiones.

Las otras dos líneas de datos, Result_ALU y DIR_WB son solo líneas que se dirigen desde la ALU hasta WB, pero que no tienen ninguna implicación sobre la memoria de la microarquitectura.

b. Control

Existe una única bandera llamada MEM_WR. Esta es la encargada de comunicarle a la memoria que la operación es de escritura y no de lectura.

6. Etapa de escritura (*Write Back*)

En la figura 6.1, se muestra la organización de la etapa de escritura.

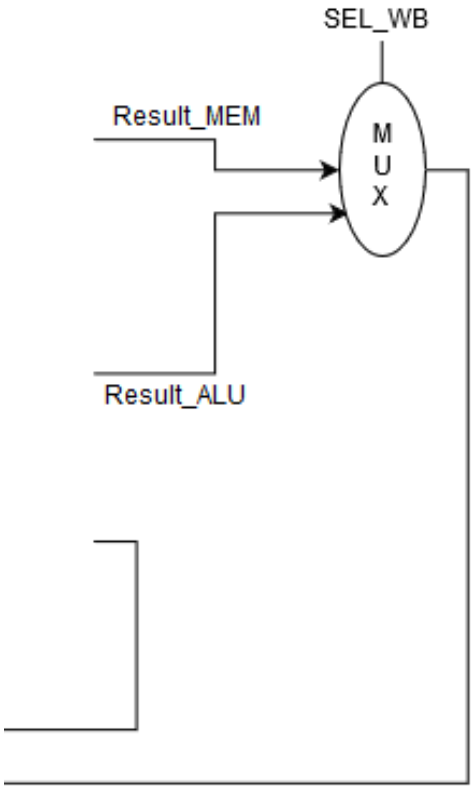

Figura 6.1. Etapa de escritura de la microarquitectura.

a. Ruta de datos

Dentro de esta etapa existe un multiplexor con tres entradas: una entrada la cual es el dato que proviene de la memoria (Result_MEM) y la otra entrada es el resultado de la operación realizada por la ALU (Result_ALU), sin haber pasado por la etapa de memoria. La salida será alguna de las dos mencionadas anteriormente, y por ende, se escribirá este dato en el banco de registros, en la dirección correspondiente.

b. Control

La señal restante SEL_WB es la que controla el comportamiento del multiplexor. De esta manera, según el valor que tome esta señal, la salida será alguna de las dos entradas mencionadas anteriormente.

7. Diagrama completo

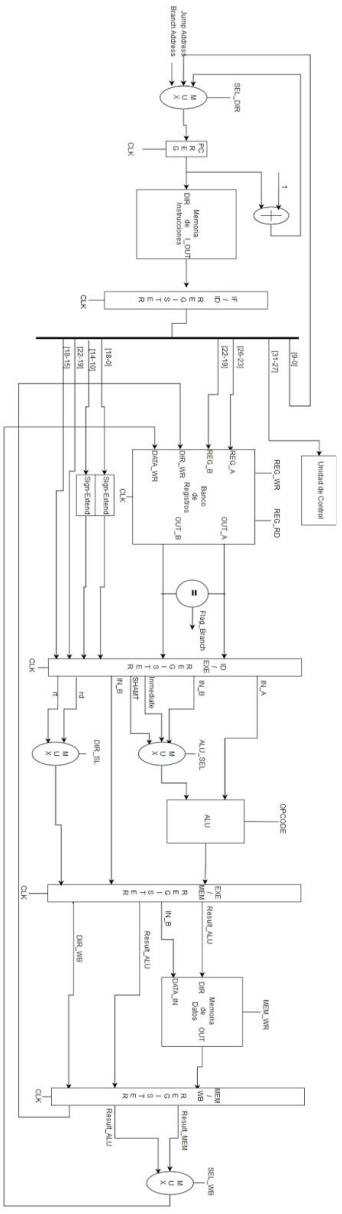

Figura 7.1. Diagrama completo de la microarquitectura.

8. Decisiones tomadas

- El hardware que compone el procesador fue diseñado a partir del MIPS, aunque muchos de los componentes del mismo no eran necesarios en el procesamiento de las instrucciones para el análisis de texto, por lo cual se removieron. De este modo, al disminuir hardware, se disminuye el costo, el área y la potencia necesaria para el mismo.
- El PC consta de diez bits, por lo que el área, la potencia y el costo disminuyen, ya que hay menor cantidad de buses de datos.
- Se cuenta con dos memorias: una para datos y otra para instrucciones. Esto aumenta el rendimiento y elimina los riesgos estructurales. Además de que permite la separación modulada de los datos e instrucciones.
- Para la instrucciones *branch*, se realiza la comparación de registros en la etapa de decodificación para poder obtener más rápidamente el resultado del mismo con el fin de agilizar la ejecución de este tipo de instrucciones.
- Al tener pocas instrucciones, al mismo tiempo se tienen menor cantidad de banderas de control, por lo que se requieren menor cantidad de multiplexores u otros componentes utilizados, y esto genera una disminución en el costo, área y potencia.
- El PC se aumenta en una unidad para acceder a las instrucciones, ya que la estructura interna de la memoria de instrucciones está establecida de esa forma, por su addressability se sabe que dentro de una celda habrá una instrucción completa.
- Se decidió agregar un multiplexor en la etapa de memoria, con el objetivo de seleccionar entre los 32 bits que componen un dato, o bien, 1 byte de este dato, es decir, 8 bits. Esto con el fin de obtener un caracter y almacenarlo en memoria.
- La microarquitectura cuenta con un control de saltos, el cual realiza un manejo de saltos adecuado. Si se cumple la condición de salto, este lo verifica y posteriormente lo realiza. Este componente se encuentra en la etapa de búsqueda o *fetch*.

9. Tabla explicativa

Elementos del ISA	JOF32
Clase de ISA	• RISC: Se decidió esta clase ya que las instrucciones son más simples, y por ende, la microarquitectura se puede simplificar. • Load/Store: Para acceso a memoria únicamente se utilizan las operaciones de carga y almacenamiento de datos, esto permite que la microarquitectura sea más ágil ya que la mayoría de instrucciones utilizan registros.
Registros	• El procesador cuenta con 16 registros de 32 bits en total de los cuales: 14 son de uso general, 1 registro para almacenar el número cero, 1 registro para la bandera de salto. Se decidió utilizar solo 16 debido a que, para la tarea específica, eran los necesarios para cumplir con esta.
Tipos de datos	• Bytes: Esto debido a que cada caracter a analizar se representa con un byte. • Palabras de 4 bytes: Esto con el fin de almacenar datos completos en las celdas de memoria, ya que cada celda es de 32 bits.
Modos de direccionamiento	• Inmediato con desplazamiento *[Rn, #±imm]: Para las instrucciones de formato I.* • Registro *[Rn]: Para las instrucciones de formato R.* • Registro escalado con desplazamiento *[Rn, ±Rm, shift]: Para las instrucciones de formato R y J.*
Organización de memoria	• Soporta Big-Endian, por cuestiones de comodidad. • Espacio de direccionamiento: 2MB. Se decidió así para soportar las cadenas de texto, así como los patrones. • Addressability: 32 bits. Esto con el fin de almacenar palabras completas dentro de una celda de memoria. • Todos los accesos a memoria deben estar alineados.

	● Los accesos a memoria pueden ser a nivel de byte y palabra. Esto con el fin de manipular de manera más fácil los caracteres que componen las cadenas de texto para realizar los respectivos algoritmos.
Formatos de instrucción	● Data processing immediate shift ● Data processing register shift ● Data processing immediate ● Load/Store Immediate offset ● Load/Store register offset ● Load/Store multiple ● Branch/Branch with link Se escogieron los formatos anteriores debido a que, para nuestro set de instrucciones necesarios para realizar los algoritmos, son los formatos más pertinentes a utilizar.
Operaciones	● Load/Store. ● Operaciones de ALU (lógicas y aritméticas). ● Branches. ● Saltos. Se decidió implementar este conjunto de operaciones para llevar a cabo la implementación de los algoritmos de forma exitosa.
Codificación	● Instrucciones de tamaño fijo: 32 bits. (**Véase el documento de descripción de ISA**) ● El modo de direccionamiento varía según la instrucción, y cada una de estas posee un código de operación único.

10. Flujos de instrucciones

Se ha anexado archivo en formato pdf.