Departamento de Ingeniería de Sistemas
Arquitectura de Computadores y Laboratorio – 2020-1
Laboratorio No. 4
Diseño del procesador monociclo

UNIVERSIDAD DE ANTIOQUIA
1803
Facultad de Ingeniería

Fecha de asignación: 23 de noviembre de 2020
Fecha límite de entrega: 23:59 horas del 18 de diciembre de 2020

1. Objetivo

- Comprender los requerimientos de una versión reducida de la arquitectura MIPS de 32 bits para realizar una implementación en forma de procesador monociclo (ruta de datos y unidad de control).
- Codificar, ensamblar y simular un programa de prueba para verificar el comportamiento correcto del procesador.
- Emplear herramientas de software para el diseño y la simulación de computadores digitales.

2. Descripción

En esta práctica cada equipo realizará una implementación como procesador monociclo de una versión reducida de la arquitectura MIPS32 en la herramienta ***Logisim Evolution***. Una vez implementados la ruta de datos y la unidad de control, el procesador será puesto a prueba para verificar el correcto funcionamiento de todas las instrucciones elegidas. Luego, el procesador será empleado para ejecutar un programa, cargado en la memoria de instrucciones en código de máquina. El programa por ejecutar corresponderá a la solución de un problema planteado en la Sección 4.

El procesador debe estar en capacidad de ejecutar las siguientes instrucciones: **`load word`**, **`store word`**, **`add`**, **`sub`**, **`and`**, **`or`**, **`nor`**, **`set-on-less-than`**, **`branch if equal`**, **`jump`**, **`jump-and-link`** y **`jump-register`**, y tener una organización como la que se muestra en la Figura 1. Los formatos de instrucción con sus códigos de operación, identificadores de registros y valores del campo función se deben ajustar a los definidos en el documento **MIPS Reference Data**. El control de la ruta de datos se debe realizar con los módulos "Control" y "ALU control" incluidos en la Figura 1, aplicando las técnicas de diseño combinacional estudiadas previamente en el curso.

El diseño de la ALU debe seguir un estilo estructural y jerárquico, en el que se diseñan bloques básicos que luego son instanciados para crear otros más complejos y de mayor nivel en la jerarquía de diseño. Para construir la ALU, de la biblioteca de componentes de ***Logisim Evolution*** sólo se podrán emplear los siguientes componentes: compuertas AND, OR, NOT y multiplexores. La estructura jerárquica básica a nivel de bloques de diseño que debe tener la ALU es la siguiente:

- ALU 32 bits
  - ALU 1 bit
    - Sumador completo de 1 bit

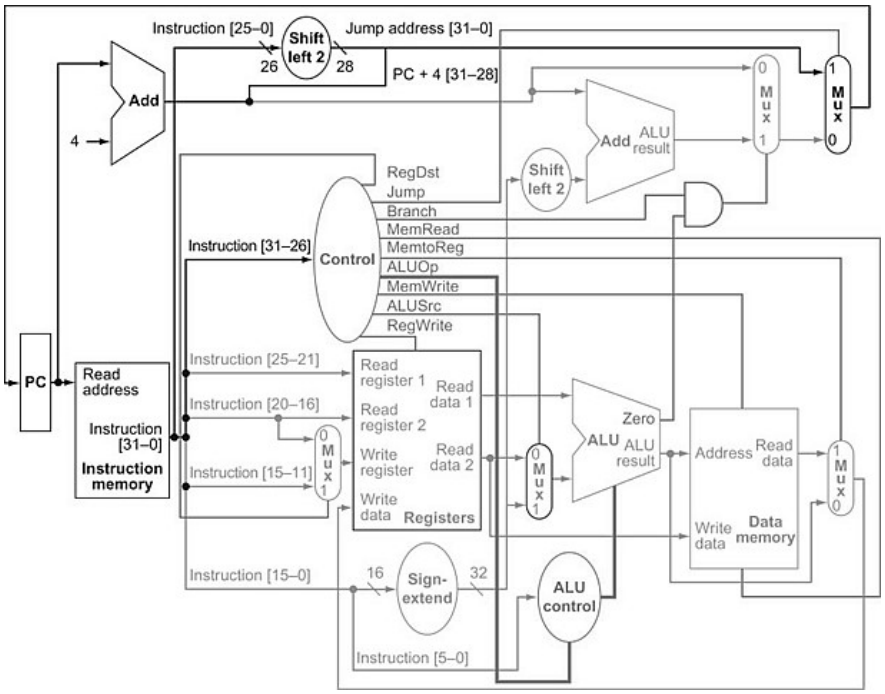

Figura 1. Implementación monociclo de la arquitectura MIPS capaz de ejecutar instrucciones lw, sw, add, sub, and, or, nor, slt, slr, sll, beq y jump (no incluye las instrucciones jal y jr).

La codificación de las señales de control asociadas a la ALU, ilustradas en la Figura 1, que debe implementar cada equipo es asignada en la Sección 3 de esta guía.

El diseño del banco de registros se debe realizar siguiendo un estilo jerárquico, partiendo de un registro de 32 bits disponible en la biblioteca de componente de ***Logisim Evolution***, que será instanciado tantas veces como sea necesario para conformar el banco. Los puertos de lectura y escritura pueden ser implementados usando decodificadores y multiplexores disponibles en la misma biblioteca. El banco de registros debe cumplir con las siguientes condiciones: (i) la lectura de los registros debe ser asíncrona, mientras que la escritura debe ser síncrona; y (ii) de acuerdo con la arquitectura del conjunto de instrucciones MIPS de 32 bits, el registro **`$zero`** siempre debe entregar el valor cero cuando sea accedido en modo de lectura, y su valor no puede ser modificado mediante una operación de escritura.

Para las memorias, registros, sumadores, multiplexores y demás componentes que sean requeridos, puede emplear componentes de la biblioteca de la herramienta.

Durante el proceso de diseño e implementación del procesador pueden presentarse diversas situaciones ante las cuales cada equipo debe tomar decisiones. En todos los casos, las decisiones de diseño deben estar ampliamente explicadas y justificadas.

3. Control de la ALU

La codificación de las operaciones de la ALU que debe implementar cada equipo se calcula como (*Número del equipo* % 10). Las codificaciones son las siguientes:

Codificación #0

Función	**ALU Operation**
AND	011
OR	001
NOR	000
Suma	101
Resta	110
SLT (*Set on less than*)	100

Codificación #1

Función	**ALU Operation**
AND	010
OR	111
NOR	011
Suma	001
Resta	110
SLT (*Set on less than*)	101

Codificación #2

Función	**ALU Operation**
AND	010
OR	011
NOR	111
Suma	101

Resta	001
SLT (*Set on less than*)	100

Codificación #3

Función	**ALU Operation**
AND	110
OR	111
NOR	101
Suma	100
Resta	001
SLT (*Set on less than*)	000

Codificación #4

Función	**ALU Operation**
AND	000
OR	100
NOR	001
Suma	011
Resta	110
SLT (*Set on less than*)	111

Codificación #5

Función	**ALU Operation**
AND	101
OR	001
NOR	011
Suma	000
Resta	100
SLT (*Set on less than*)	010

Codificación #6

Función	ALU Operation
AND	011
OR	001
NOR	111
Suma	110
Resta	101
SLT (*Set on less than*)	100

Codificación #7

Función	ALU Operation
AND	000
OR	110
NOR	100
Suma	011
Resta	001
SLT (*Set on less than*)	111

Codificación #8

Función	ALU Operation
AND	100
OR	110
NOR	101
Suma	001
Resta	000
SLT (*Set on less than*)	010

Codificación #9

Función	ALU Operation
AND	011
OR	001

NOR	000
Suma	101
Resta	110
SLT (*Set on less than*)	100

4. Programas de prueba

   A cada equipo de trabajo le corresponderá escribir un programa en ensamblador MIPS, y obtener el respectivo código de máquina, que resuelva uno de los problemas listados a continuación. El problema asignado a cada equipo se calcula como (*Número del equipo* % 8).

   Los requisitos a los que se debe ajustar la codificación en ensamblador son los siguientes:

   - Programación basada en el uso de procedimientos. El código que resuelve el problema asignado debe ser escrito como un procedimiento, el cual será invocado desde el programa principal.
   - Seguir la convención para el uso de registros MIPS
   - Se valorará la creatividad y elegancia en la programación.

   Problema #0
   El programa debe tomar un vector de cien números enteros que contiene valores entre -100 y 100, y determinar cuáles son el mayor y el menor número almacenados, y cuántas veces se repite cada uno.

   Problema #1
   El programa debe tomar un vector de cien números enteros que contiene valores entre -100 y 100, y determinar cuáles son el mayor y el menor número almacenados entre los elementos de índice par e impar, y cuántas veces se repite cada uno.

   Problema #2
   El programa debe tomar un vector de cien números enteros que contiene valores entre -100 y 100, y ordenarlo de manera ascendente, indicando cuántos elementos son negativos y cuántos son positivos.

   Problema #3
   El programa debe tomar un vector de cien números enteros que contiene valores entre -100 y 100, y ordenarlo de manera descendente, indicando cuántos elementos son pares y cuántos son impares.

Problema #4
El programa debe tomar una matriz de 10×10 números enteros y determinar cuáles son el mayor y el menor número almacenado en cada fila y en cada columna.

Problema #5
El programa debe tomar una matriz de 10×10 números enteros y determinar cuántos números positivos y negativos hay en cada fila y columna.

Problema #6
El programa debe tomar una matriz de 10×10 números enteros y determinar cuántos números pares e impares hay en cada fila y columna.

Problema #7
El programa debe tomar una matriz de 10×15 números enteros y calcular su transpuesta multiplicada por el escalar 5.

5. Informe

Cada equipo de trabajo debe realizar un informe escrito que incluya una descripción completa del proceso de diseño del procesador monociclo (ruta de datos y unidades de control), sus esquemáticos y la explicación y justificación de las decisiones de diseño tomadas durante el desarrollo de la práctica. Asimismo, debe describir el proceso de verificación de la ejecución correcta de todas las instrucciones que soporta el procesador. También debe incluir una descripción de alto nivel (mediante pseudocódigo, diagrama de flujo, etc.) del programa implementado que da solución al problema planteado, suministrando su código fuente en ensamblador y el código de máquina equivalente. El informe también debe contener resultados de ejecución, y las observaciones y conclusiones pertinentes del trabajo. Este documento debe estar en formato **PDF** y ser subido a la plataforma antes del cierre del plazo de entrega, con los archivos de diseño en ***Logisim Evolution*** como adjuntos. El informe tiene un peso del 30% en la calificación global de la práctica.

6. Sustentación

Ambos miembros del equipo de trabajo deben demostrar un dominio completo del desarrollo de la práctica. El profesor planteará preguntas para evaluar los conocimientos adquiridos (diseño de la ruta de datos y las unidades de control; algoritmos empleados e implementación del programa; fundamentos conceptuales de la arquitectura del conjunto de instrucciones MIPS: formatos de instrucción, convención de uso de registros MIPS, uso del *stack pointer*, programación basada en procedimientos, entre otras) La sustentación tiene un peso del 70% en la calificación global de la práctica.