MIPS SEGMENTADO Y GESTIÓN DE RIESGOS

ARQUITECTURA Y ORGANIZACIÓN DE COMPUTADORES 2

DANIEL RUEDA MACÍAS

NIP: 559207

 

Índice

[RESUMEN 3](#_Toc448396702)

[INTRODUCCIÓN 3](#_Toc448396703)

[OBJETIVOS 3](#_Toc448396704)

[METODOLOGÍA 4](#_Toc448396705)

[LW y SW con preincremento 4](#_Toc448396706)

[Gestión de riesgos de datos 4](#_Toc448396707)

# RESUMEN

En este proyecto se ha optimizado un MIPS segmentado añadiéndole instrucciones preincremento, paradas en la detección de riesgos de datos y un predictor de saltos.

Para las instrucciones preincremento, se ha añadido un nuevo puerto de escritura junto con una señal de control independiente a la que ya tenía; para las paradas de detección de riesgos de datos, se ha condicionado las señales de control del PC y del banco que separa las etapas de fetch y decode si se detectaba alguno de estos riesgos; y por último, a la hora de añadir el detector de saltos, se ha ampliado el mux del PC a un mux de cuatro entradas con dos bits de control y se ha añadido el precitor correspondiente conectándole como entradas la información del útimo salto realizado y como salidas la información del salto predicho.

Todo esto con el fin de reducir los CPIs y en consecuencia, el tiempo de ejecución del procesador para así poder obtener uno mucho más eficiente.

# INTRODUCCIÓN

Se nos ha proporcionado un MIPS segmentado de 32 bits como el visto en clase sin anticipación de operandos, que resuelve los saltos en la etapa decode y escribe en el banco de registros en flanco de bajada. Todo descrito mediante VHDL, un lenguaje de descripción de hardware y ModelSim, un entorno de simulación basado en bancos de prueba y cronogramas.

# OBJETIVOS

Queremos reducir los CPIs del MIPS, mediante el añadido de instrucciones preincremento, un módulo que detecte los riesgos de datos realizando las paradas necesarias y un predictor de saltos que diga si se va a saltar o no en la etapa fetch según lo que se ha hecho en el salto anterior.

# METODOLOGÍA

## LW y SW con preincremento

Para la realización de estas instrucciones preincremento, lo primero que se ha hecho ha sido modificar el banco de registros, añadiéndole un nuevo puerto de escritura, junto con un bus y una señal de control.

Lo segundo ha sido propagar la señal correspondiente al registro RS hasta la etapa MEM. En dicha etapa, se ha conectado la señal ALU\_out\_MEM, que es la que contiene la salida de la ALU, al nuevo bus del banco de registros. Lo mismo ha sucedido con RS, en dicha etapa, se ha conectado al nuevo puerto de escritura del banco.

Por último pero no menos importante, se ha añadido una nueva señal a la unidad de control, que es la que corresponde a la escritura en el banco de registros de las instrucciones preincremento, poniéndola a 1 sólo cuando el código de operación sea el correspondiente a las instrucciones LW\_pre o SW\_pre.

## Gestión de riesgos de datos

Mediante distintos ejemplos de código pensados en papel realizando los grafos de dependencias, se ha llegado a la detección de los siguientes riesgos de datos.

* Riesgos de datos de RS
  + Que en la etapa EX el registro que haya en el puerto de escritura del banco (RW\_EX) coincida con el nuestro y que se vaya a realizar la escritura (RegWrite\_EX = 1).
  + Que en la etapa MEM el registro que haya en el puerto de escritura del banco (RW\_MEM) coincida con el nuestro y que se vaya a realizar la escritura (RegWrite\_MEM = 1).
  + Que en la etapa EX el registro que haya en el puerto de escritura que se ha añadido para la instrucciones preincremento (RS\_EX) coincida con el nuestro y se vaya a realizar la escritura (Update\_Rs\_EX = 1).
* Riesgos de datos para RT
  + Que en la etapa EX el registro que haya en el puerto de escritura del banco (RW\_EX) coincida con el nuestro, que se vaya a realizar la escritura (RegWrite\_EX = 1) y que la instrucción que se encuentra en D no sea un LW.
  + Que en la etapa MEM el registro que haya en el puerto de escritura del banco (RW\_MEM) coincida con el nuestro, que se vaya a realizar la escritura (RegWrite\_MEM = 1) y que la instrucción que se encuentra en D no sea un LW.
  + Que en la etapa EX el registro que haya en el puerto de escritura que se ha añadido para la instrucciones preincremento (RS\_EX) coincida con el nuestro y se vaya a realizar la escritura (Update\_Rs\_EX = 1) y que la instrucción que se encuentra en D no sea un LW.

Éstos riesgos se deben considerar para todas las instrucciones excepto para una NOP, una NOP no hace nada, aunque tenga operandos, por lo tanto no debemos hacer parada. En cuanto al comentario que se ha realizado respecto a que la instrucción no sea un LW en los riesgos de RT, esto se debe a que en un LW en RT se carga el dato, no se lee de él, por lo tanto no tenemos que parar.

## Gestión de riesgos de control