66:20 Organización de computadoras
Trabajo práctico 3: Data path y pipeline.

1. Objetivos

El objetivo de este trabajo es familiarizarse con la arquitectura de una CPU MIPS, específicamente con el datapath y la implementación de instrucciones. Para ello, se deberán agregar instrucciones a diversas configuraciones de CPU provistas por el simulador DrMIPS [1]

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes.

Además, es necesario que el trabajo práctico incluya (entre otras cosas, ver sección 8), la presentación de los resultados obtenidos, explicando, cuando corresponda, con fundamentos reales, las causas o razones de cada resultado obtenido.

El informe deberá respetar el modelo de referencia que se encuentra en el grupo[1], y se valorarán aquellos escritos usando la herramienta TEX / LATEX.

4. Recursos

Usaremos el programa DrMIPS [1] para configurar y simular el data path de un procesador MIPS [4], tanto uniciclo como multiciclo.

5. Descripción.

5.1. Introducción

El programa DrMIPS nos permite evaluar distintos diseños de datapath para procesadores MIPS32, al darnos la posibilidad de organizarlo como queramos. Si bien sólo puede haber uno de algunos de los componentes del DP (como el registro de PC o la unidad de control), podemos poner sumadores, multiplexores, extensores de signo y conexiones arbitrariamente. También es

[1] `http://groups.yahoo.com/group/orga6620`

posible modificar el conjunto de instrucciones. Además de la estructura lógica del DP, DrMips nos permite escribir programas simples y simular su ejecución en el DP, mostrando los valores que toman las diversas entradas y salidas de cada elemento. El programa se puede conseguir en `https://brunonova.github.io/drmips/`, o se puede descargar para Ubuntu, ya sea desde el repositorio de Ubuntu o autorizando un repositorio externo (ver [2]) si la versión que reporta el manejador de paquetes es inferior a 2.0.1.

5.2. Datapaths

El programa viene con algunos DP ya implementados, a saber:
Uniciclo:

- `unycicle.cpu`: El DP uniciclo por defecto.
- `unycicle-no-jump.cpu`: Variante más simple del DP uniciclo que no soporta la instrucción `j`.
- `unycicle-no-jump-branch.cpu`: Una variante aún más simple que no soporta `jump` ni `branch`.
- `unycicle-extended.cpu`: Una variante que soporta instrucciones adicionales, como multiplicación y división.

Multiciclo:

- `pipeline.cpu`: El DP de pipeline por defecto, implementa detección de hazards. Los DP de pipeline no soportan la instrucción `j` (salto).
- `pipeline-only-forwarding.cpu`: Variante del DP de pipeline que implementa forwarding pero no genera stalls (genera resultados incorrectos).
- `pipeline-no-hazard-detection.cpu`: Otra variante que no hace hazard detection de ninguna manera (genera resultados incorrectos).
- `pipeline-extended.cpu`: Una variante que soporta instrucciones adicionales, como multiplicación y división, como `unycicle-extended.cpu`.

5.3. Instrucciones a implementar

A continuación se describen las instrucciones a implementar en el data path. Algunas de estas existen como instrucciones de MIPS pero no están implementadas; otras no.

- `andi Rs, Rt, Imm` (And immediate). Esta instrucción de tipo `I` carga en `Rs` el resultado de hacer un `AND` entre el contenido del registro `Rt` y el valor de 16 bits `Imm`.
- `j Rs, Rt` (Jump Rs+Rt). Esta instrucción de tipo `I` carga en el PC el resultado de sumar los contenidos de los registros `Rs` y `Rt`.
- `lw Rs, Rd*Imm(Rt)`. Esta instrucción de tipo `R` carga en el registro `Rs` la palabra de 32 bits cuya dirección es `Rt` + `Rd` * `shamt`. Esto resulta en el agregado del modo de direccionamiento escalado, que MIPS no tiene nativamente.

5.4. Tareas a realizar

1. Implementar la instrucción `andi Rs, Rt, Imm` en el DP `unicycle.cpu`.
2. Implementar la instrucción `andi Rs, Rt, Imm` en el DP `pipeline.cpu`.
3. Implementar la instrucción `j Rs, Rt` en el DP `unicycle.cpu`.
4. Implementar la instrucción `j Rs, Rt` en el DP `pipeline.cpu`. Verificar que no se produzcan hazards.
5. Implementar la instrucción `lw Rs, Rd*Imm(Rt)` en el DP `unicycle.cpu`.

6. Implementación.

Los archivos antes mencionados, así como los archivos `.set` que contienen los datos del conjunto de instrucciones, están en formato JSON [3], y se pueden modificar con un editor de texto. Se sugiere uno que pueda hacer *color syntax highlighting*, como el `gedit` que viene con el Ubuntu. La explicación de los formatos se encuentra en el archivo `configuration-en.pdf` que se distribuye con el programa.

7. Pruebas

En todos los casos debe verificarse que la instrucción se ejecute correctamente. Esto implica que el registro de destino tome el valor deseado, y además que en el caso del DP multiciclo no se produzcan hazards, como ser la ejecución de la instrucción siguiente al salto, o en el caso de utilizar el valor de un registro, que éste tenga el valor correcto.

8. Informe.

Se debe entregar:

- Informe describiendo el desarrollo del trabajo práctico.
- Capturas de pantalla de los DP modificados.
- Programas de prueba.
- Archivos de los DP, los programas de prueba y los conjuntos de instrucciones usados en cada caso.
- Este enunciado.

9. Fecha de entrega.

La entrega de este trabajo práctico debe realizarse el Jueves 6 de Diciembre de 2018.

Referencias

[1] DrMIPS, `https://brunonova.github.io/drmips/`.

[2] PPA de Bruno Nova, `https://launchpad.net/~brunonova/+archive/ubuntu/ppa`.

[3] ECMA-404 The JSON Data Interchange Standard, `http://www.json.org/`.

[4] "Computer organization and design: the hardware-software interface", John Hennessy, David Patterson. Capítulo 5.