

# 66:20 Organización de computadoras

## Trabajo práctico 2: 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<sup>1</sup>, y se valorarán aquellos escritos usando la herramienta  $\text{\TeX}$  /  $\text{\LaTeX}$ .

### 4. Recursos

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

### 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

---

<sup>1</sup><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:

- `uncycle.cpu`: El DP uniciclo por defecto.
- `uncycle-no-jump.cpu`: Variante más simple del DP uniciclo que no soporta la instrucción `j`.
- `uncycle-no-jump-branch.cpu`: Una variante aún más simple que no soporta `jump` ni `branch`.
- `uncycle-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 `uncycle-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.

- `lbu Rs, Imm(Rt)` (Load byte unsigned). Esta instrucción de tipo I carga en `Rs` el valor del byte almacenado en la posición resultante de sumar el valor del registro `Rt` y el valor de 16 bits `Imm`.
- `sb Rs, Imm(Rt)` (Store byte). Esta instrucción de tipo I almacena el byte menos significativo de `Rs` en la posición resultante de sumar el valor del registro `Rt` y el valor de 16 bits `Imm`.
- `bgezal Rs, Imm`. Esta instrucción de tipo I compara el valor del registro `Rs` con cero, y si es mayor o igual salta a la posición  $(PC + 4 + Imm * 4)$ , almacenando el valor de  $PC + 4$  en el registro `ra`.

---

## 5.4. Tareas a realizar

1. Implementar la instrucción `lbu R1, Imm(R2)` en el DP `unicycle.cpu`.
2. Implementar la instrucción `sb R1, Imm(R2)` en el DP `pipeline.cpu`. Verificar que no se produzcan errores por hazards de datos.
3. Implementar la instrucción `bgezal R1, Imm` en el DP `pipeline.cpu`. Verificar que no se produzcan errores por hazards de control o de datos.

## 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 los hazards se resuelvan correctamente, 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.
- CD conteniendo los DP, los programas de prueba y los conjuntos de instrucciones usados en cada caso.
- Este enunciado.

## 9. Fechas de entrega.

- Primera entrega: Jueves 17 de Noviembre.
- Revisión: Jueves 24 de Noviembre.
- Vencimiento: Jueves 1 de Diciembre.

---

## 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.