# Trabajo Práctico 3
# Data path y pipeline

Lucas Medrano, *Padrón Nro. 99247*
lucasmedrano97@gmail.com
Federico Álvarez, *Padrón Nro. 99266*
fede.alvarez1997@gmail.com

Grupo Nro. - 2do. Cuatrimestre de 2018
66.20 Organización de Computadoras
Facultad de Ingeniería, Universidad de Buenos Aires

**Resumen**

El objetivo de este trabajo es implementar algunas instrucciones en dos configuraciones de una CPU en un emulador de MIPS32: CPU uniciclo, y CPU pipeline. Para implementar las instrucciones se deberá modificar tanto el set de instrucciones como el datapath de la CPU.

# Índice

# 1. Desarrollo

El trabajo se divide en dos partes:

1. Implementar las instrucciones en una CPU uniciclo.
2. Implementar las instrucciones en una CPU pipeline.

Las instrucciones que se implementaron son las siguientes:

1. 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.
2. 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.
3. lw Rs, Rd*shamt(Rt). Esta instrucción de tipo R carga en el registro Rs la palabra de 32 bits cuya dirección es Rt + Rd * shamt. Esta última sólo se implementara en la cpu uniciclo.

## 1.1. Implementación

### 1.1.1. Programa

Para realizar el tp se usó el emulador DrMIPS, el cual permite visualizar el datapath que se usa, escribir programas, y seguir tanto el recorrido del programa y las señales en el datapath, como los valores de los registros. Modificando los archivos .cpu se modificaron ambos datapaths (unicycle y pipeline), y modificando los archivos .set se adecuaron ambos set de instrucciones para poder implementar las nuevas instrucciones.

### 1.1.2. andi - CPU uniciclo

Para esta instrucción no hizo falta agregar ninguna estructura nueva al datapath. Generando un nuevo codigo de operacion (op) y seteando las señales de control de la Unidad de Control y de la ALU se pudo obtener el resultado deseado.

### 1.1.3. andi - CPU pipeline

Al igual que en la versión uniciclo, para implementar esta instrucción no hizo falta más que generar un nuevo `opcode`, indicando las señales de control necesarias. En este caso se seteo `ALUOp` en el valor correspondiente a la operación `and`, y `ALUSrc` se seteó a 1 para indicar que en la segunda entrada de la ALU debe ingresarse el valor del campo `imm`. No fue necesario modificaar nada para el manejo de *hazards* ya que con el soporte ya proporcionado bastaba.

### 1.1.4. jregs - CPU uniciclo

Para poder implementar esta instrucción no bastaban las estructuras que incluía el datapath uniciclo. Se agregó un multiplexor que, mediante una entrada de control, decide si la siguiente direccion de PC es el resultado de la operación de la ALU o es la que indica MuxjUMP (PC+4/branch addres o jump address). Así, se puede setear que se sumen los registros indicados por parámetro, y que la siguiente direccion del PC sea dicha suma.

### 1.1.5. j - CPU pipeline

Para implementar `jump` se agregó un multiplexor que toma la salida del multiplexor que maneja los branches y la salida de la ALU. Para controlarlo, se agregó una señal de control llamada *Jump*, que se setea únicamente por esta operación, y debió ser pasada a través de las distintas etapas del pipeline, lo que requirió agregar un bit más a los registros ID/EX y EX/MEM.

Para el manejo de hazards se agregó una compuerta `Or` que toma las señales de Branch y Jump y se las conecta a los registros de cada etapa del pipeline en la entrada *Flush*, para agregar un ciclo de stall cuando se ejecuta un salto.

### 1.1.6. lw - CPU uniciclo

Para implementar esta instrucción se tomaron algunas decisiones:

1. No modificar el datapath: Ya que hay que hacer una multiplicación y luego obtener el resultado del registro "LO"mediante la instrucción mflo, se optó por dejar el datapath igual, e implementar la instrucción como una pseudo-instrucción.

2. Implementarla como lw r1, r2, shamt, r3, debido a la sintaxis de MIPS.

Se implementó la instrucción como una pseudo-instrucción: se multiplicó el valor de shamt por el de r2, se le sumó el de r3, se pidió el dato de memoria de dicha instrucción y se lo almacenó en r1. Una idea fue agregar un adder, para poder hacer el producto en el mismo ciclo de reloj. Sin embargo, primero se necesitaba mover el resultado del registro "LO.ª algún registro, por lo que se optó lo indicado previamente.

## 1.2. Datapaths

Los datapath utilizados, y modificados, para el trabajo quedaron de la siguiente manera.

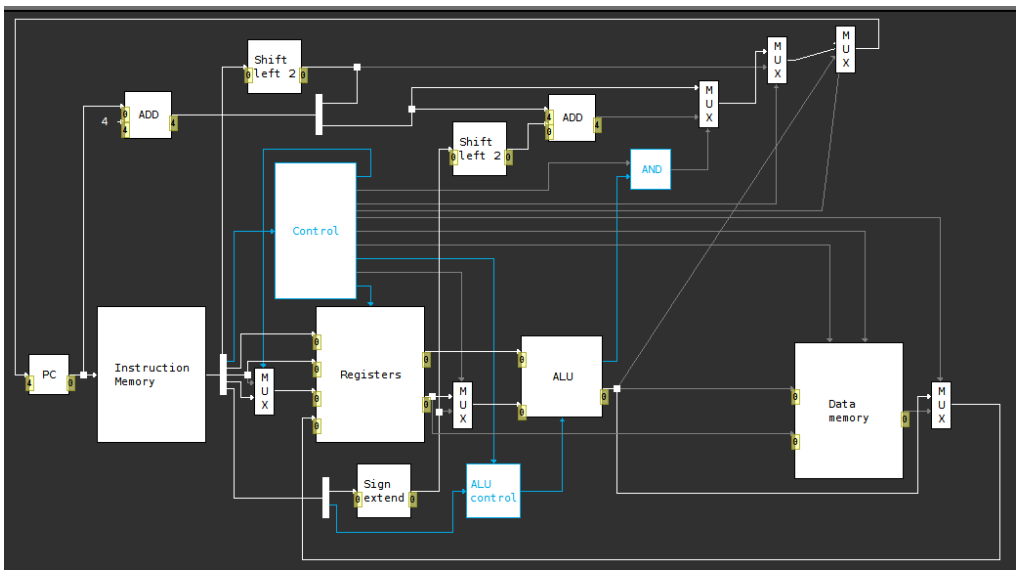


Figura 1: Datapath de la CPU-Unicycle.

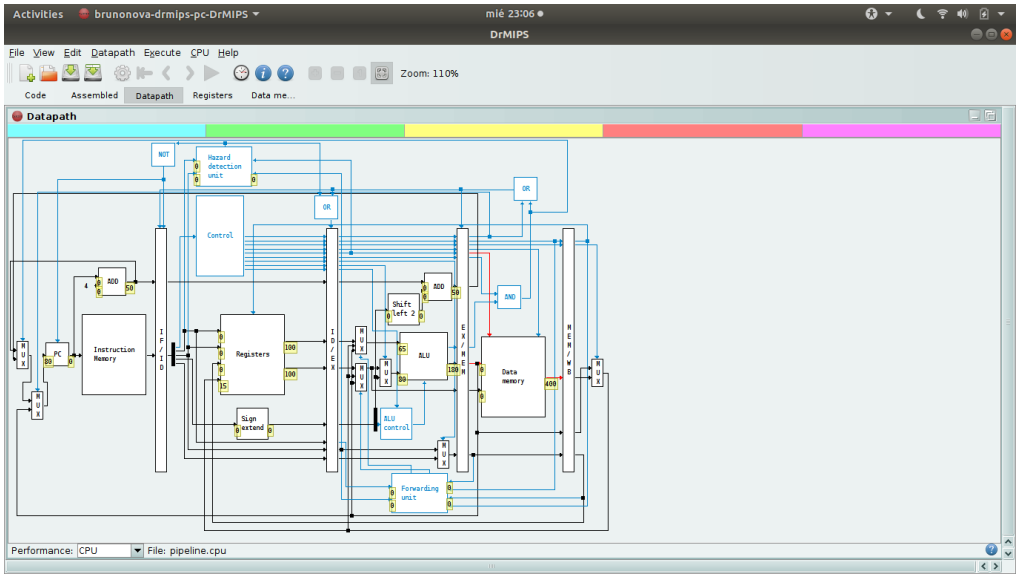


Figura 2: Datapath de la CPU-Pipelined.

## 1.3. Pruebas

Para cada instrucción implementada se agregan una, o más, pruebas para asegurarse que el comportamiento sea el esperado. Se adjuntan dichas pruebas a los archivos del trabajo.

# 2. Conclusiones

Se logró implementar las instrucciones pedidas. El trabajo nos ayudó a entender varios conceptos. Modificar el datapath, las estructuras y cableados permite entender mucho mejor el funcionamiento del procesador en general y de cada estructura en particular. Modificar los set de instrucciones ayuda a ver mucho mejor cómo influye el datapath utilizado a la hora de generar instrucciones. Aprendimos que el datapath define el set de instrucciones que se podrá utilizar, pero un mismo set de instrucciones puede ser implementado de varias maneras distintas (en este caso con un CPU uniciclo y uno Pipeline). Además, se logró conceptualizar la diferencia entre el CPU uniciclo y el CPU pipeline. EL programa DrMips fue de gran ayuda para poder visualizar no solo el datapath, los set de instrucciones, los registros en todo momento, sino también el camino que seguía cada instrucción en el datapath al ser ejecutada.

# 3. Apéndice

## 3.1. Enunciado

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.