Trabajo Práctico Nro 3
Organización de Computadoras
Datapath y Pipeline

Tomás Lubertino, *Padrón 95140*
tomas.lubertino@gmail.com

Maximiliano Suppes, *Padrón 96938*
maxisuppes@gmail.com

Zuretti Agustin, *Padrón 95605*
zurettiagustin@gmail.com

1er. Cuatrimestre de 2020
86.37 / 66.20 Organización de Computadoras − Práctica Jueves
Facultad de Ingeniería, Universidad de Buenos Aires

Resumen

El siguiente informe explica la resolución del trabajo práctico número 3 de la materia, en el cuál se implementarán las instrucciones: andi, jump y lwi utilizando el software DrMips.

Índice

1. Objetivo

El objetivo de este trabajo es familiarizarse con la arquitectura de una CPU MIPS, específicamente con el datapath (unicycle y pipeline) y la implementación de instrucciones sobre estos.

2. Introducción

En este trabajo práctico se pide implementar tres nuevas instrucciones agregandolas al .set o modificando el datapath en caso de ser necesario. Para esto se va a utilizar el software 'DrMIPS' que provee una interfaz gráfica que permite evaluar distintos diseños de datapath para procesadores MIPS32, modificando e incorporando componentes (como multiplexores, sumadores, extensores de signos, componentes Or y And) y definiendo el cableado y camino de datos para implementar nuevas instrucciones. Además se puede observar valores de entrada y salida de todos los componentes en cada instrucción del programa.

3. Desarrollo

Utilizando el programa DrMIPS, se pide la implementación de las siguientes instrucciones:

- andi Rs, Rt, Imm (And immediate). Esta instrucci on 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 instrucci on 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 instrucci on de tipo R carga en el registro Rs la palabra de 32 bits cuya direcci on es Rt + Rd *Imm. Esto resulta en el agregado del modo de direccionamiento escalado, que MIPS no tiene nativamente.

Para andi y j se debe implementar la instrucción en el DP unicycle.cpu y pipeline.cpu. Mientras que la instrucción lw se debe implementar solo en el DP pipeline-extended.cpu.

3.1. Repositorio

`https://github.com/MaxiSuppes/mips-pipelines`

3.2. Datapath unicycle y pipeline

En la siguiente figura se observa el datapath unicycle que provee la herramienta. Será necesario para poder comparar los cambios que se realizarán en los próximos puntos.

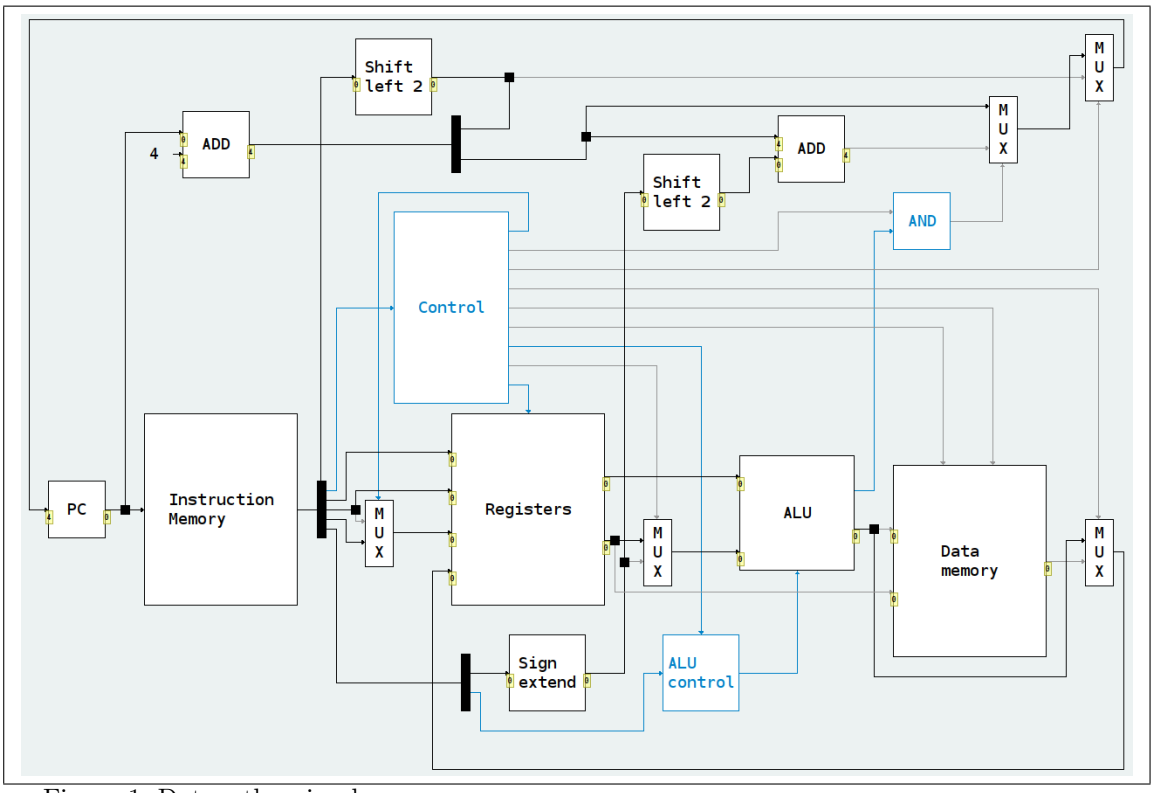

Figura 1: Datapath unicycle.

La siguiente figura muestra el datapath pipeline que provee DrMIPS. Será necesario para poder comparar los cambios que se realizarán en los próximos puntos.

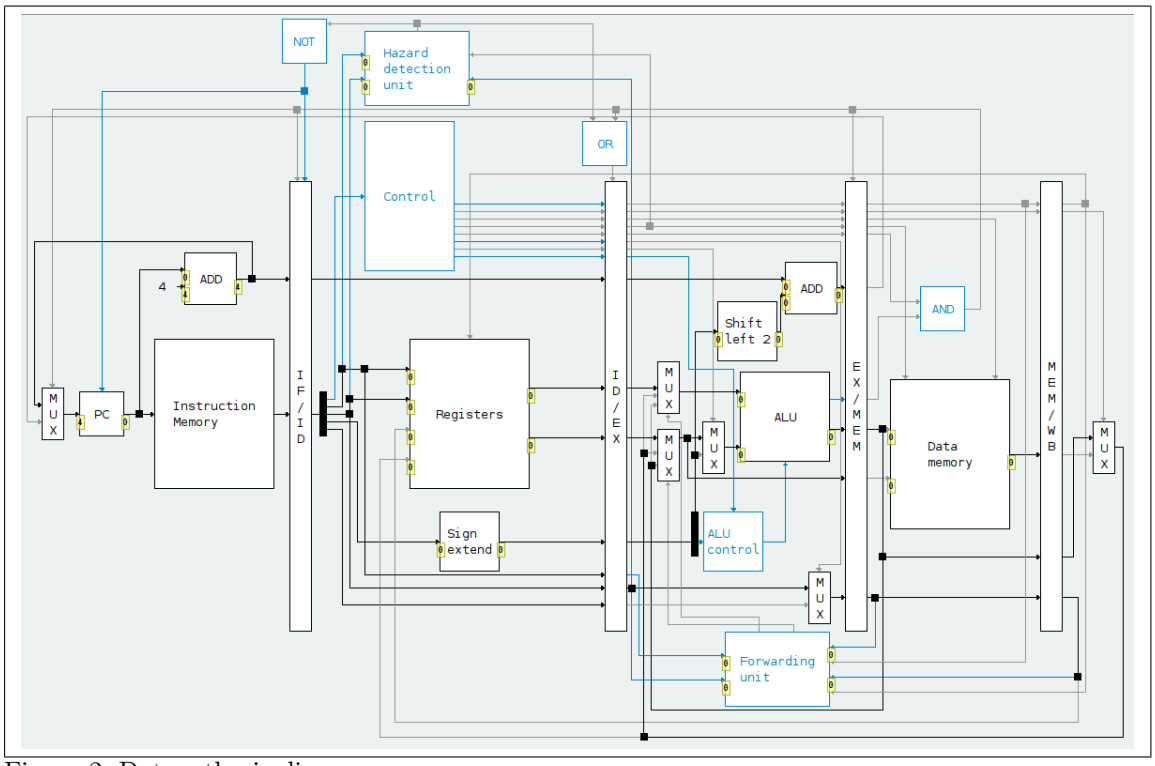

Figura 2: Datapath pipeline.

3.3. Instrucción andi en DP unicycle

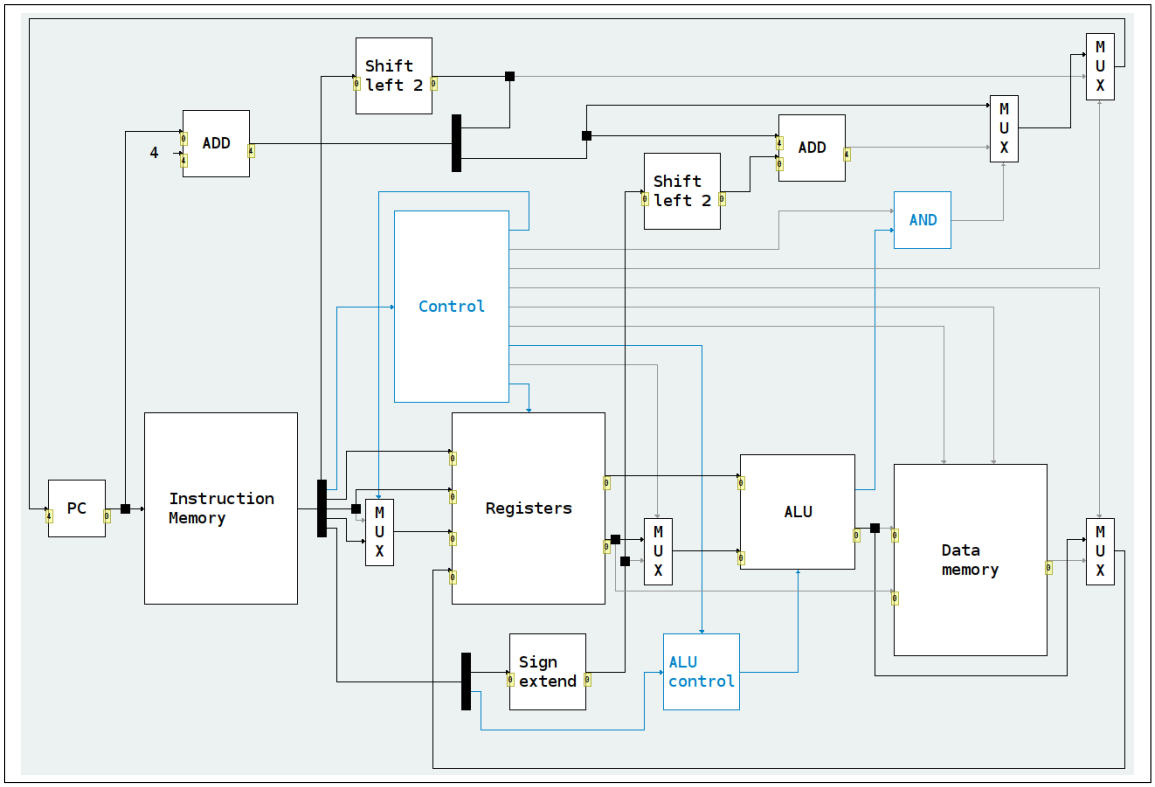

Figura 3: Datapath unicycle con la instrucción `andi` implementada.

Para implementar esta instrucción solo se agregó al set de instrucciones con su correspondiente configuración y no fue necesario modificar la arquitectura del datapath, ya que este implementa instrucciones similares.

3.4. Instrucción andi en DP pipeline

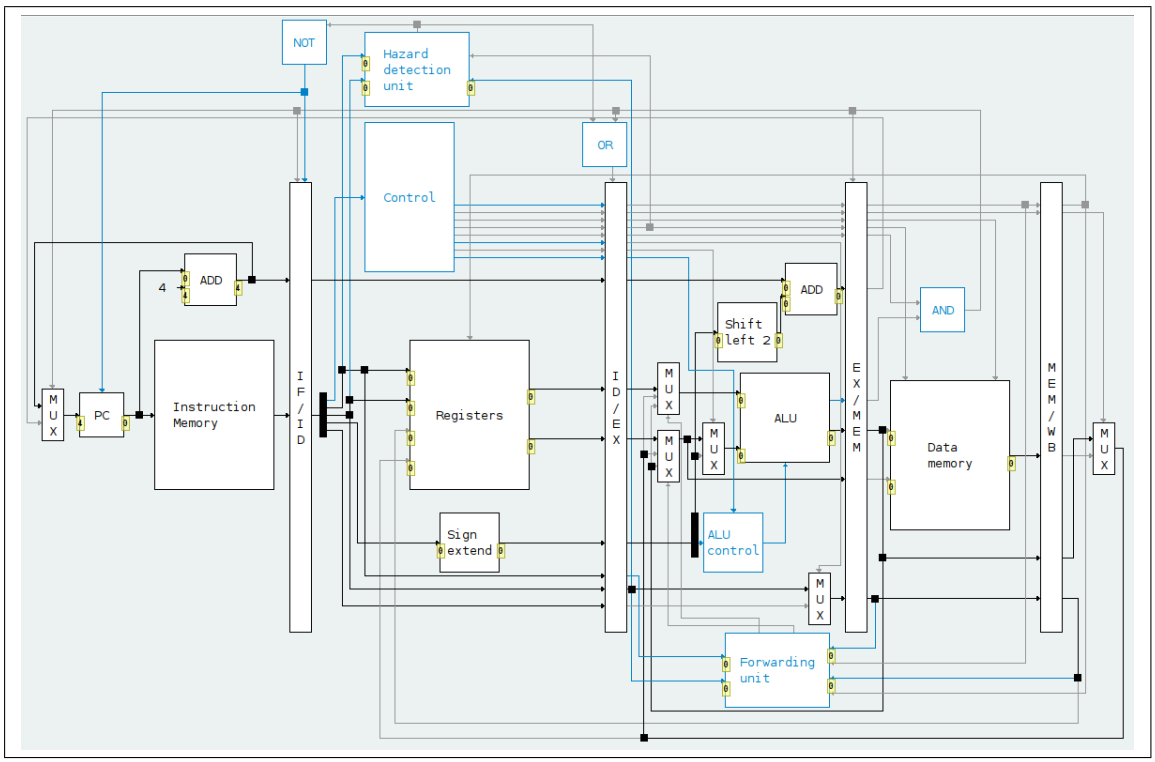

Figura 4: Datapath pipeline con la instrucción `andi` implementada.

Se implementó igual que para el DP unicycle y no fue necesario modificar el datapath.

3.5. Instrucción j en DP unicycle

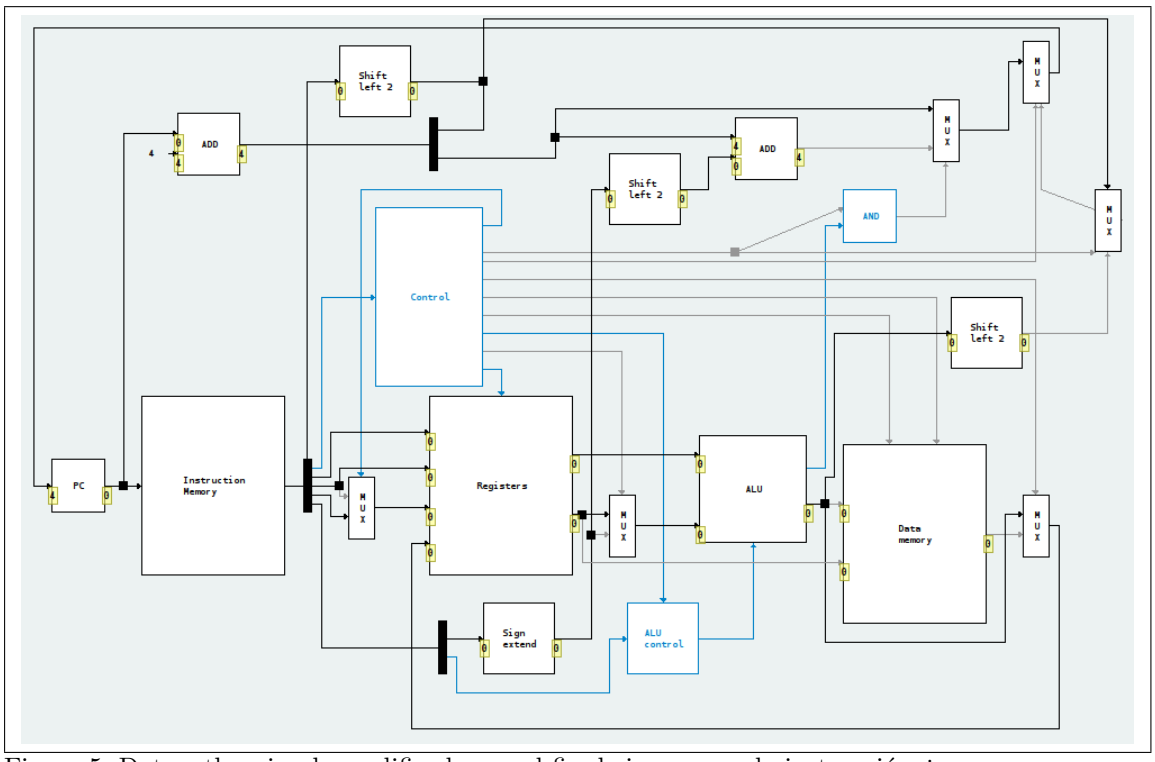

Figura 5: Datapath unicycle modificado con el fin de incorporar la instrucción `jump`.

En primer lugar se cambió el nombre de la instrucción a jump porque j ya estaba definida.Para agregar esta instrucción fue necesario modificar el datapath. Como ya se dijo, la instrucción jump ya se encuentra implementada a partir de un multiplexor llamado MuxJump que define si el nuevo valor del PC es PC+4/branch-address si el valor de la señal Jump es 0, o jump address si el valor de la señal Jump es 1. Para la nueva instrucción se agregó un multiplexor llamado MuxJumpAlu cuyas entradas son el valor de dirección que entraba al MuxJump y la salida de un Shift Left (para multiplicar por 4 que es el tamaño de una instrucción) aplicado al resultado de la ALU. La salida está definido por el valor de la señal Branch.

3.6. Instrucción j en DP pipeline

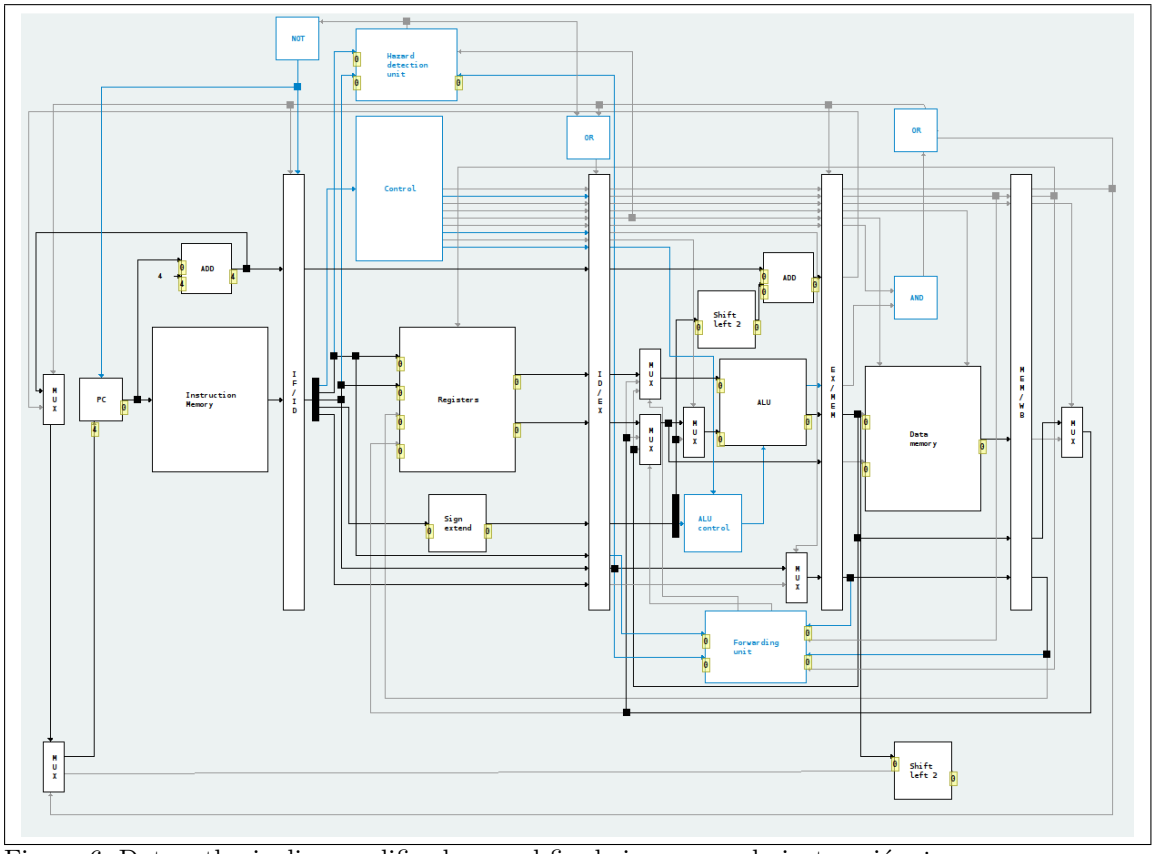

Figura 6: Datapath pipeline modificado con el fin de incorporar la instrucción `jump`.

Para incluir la instrucción en este datapath hubo que hacer algunos cambios más, ya que no tenía implementado el jump, por lo que el nuevo valor del PC era definido por un multiplexor llamado MuxPC que tenía como entradas PC+4 y la dirección de branch. Por lo tanto, hubo que agregar un nuevo multiplexor con dos entradas: La salida del MuxPc y el resultado de la ALU, tratando de imitar la lógica del unicycle para el jump. Para decidir la entrada a seleccionar se utiliza la señal Jump proveniente de la unidad de control (hubo que agregarla). Para evitar hazards es necesario que las instrucciones que ya estén dentro del pipeline cuando se está ejecutando un jump sean flusheadas. Esta lógica ya se encuentra implementada para la instrucción branch por lo cual solo fue necesario agregar un OR que combine el funcionamiento anterior con la salida del jump.

3.7. Instrucción lw en DP pipeline

Cambiamos el nombre de la instrucción a lwi ya que lw ya estaba implementada. Para la implementación de esta instrucción no fue necesario realizar cambios en el datapath. Entendiendo lo que debía realizar la instrucción, nos dimos cuenta que bastaba con agregar una nueva pseudo-instrucción que combine instrucciones que ya estaban presentes en el set. La firma de la instrucción fue levemente cambiada de `lw Rs, Rd*Imm(Rt)` a `lwi Rs, Rt, Rd, Imm` ya que Dr. Mips no permite la descomposición de un argumento data para una pseudo-instrucción.

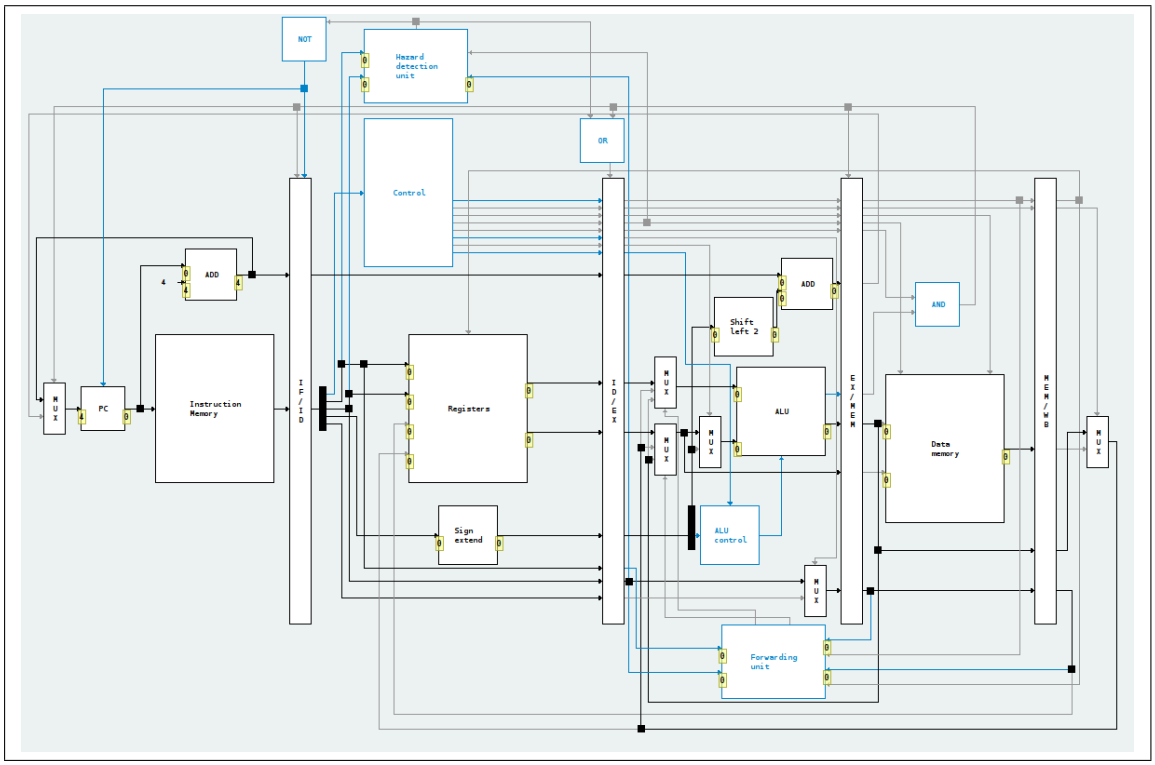

Figura 7: Datapath pipeline para la instrucción `lwi`.

3.8. Scripts de prueba

3.8.1. andi

```
1 li $t0, 1
2 andi $t1, $t0, 3
```

Register	Value
0: $zero	0
1: $at	0
2: $v0	0
3: $v1	0
4: $a0	0
5: $a1	0
6: $a2	0
7: $a3	0
8: $t0	1
9: $t1	1
10: $t2	0
11: $t3	0
12: $t4	0
13: $t5	0
14: $t6	0
15: $t7	0
16: $s0	0
17: $s1	0
18: $s2	0
19: $s3	0
20: $s4	0
21: $s5	0
22: $s6	0
23: $s7	0
24: $t8	0
25: $t9	0
26: $k0	0
27: $k1	0
28: $gp	0
29: $sp	0
30: $fp	0
31: $ra	0
PC	8

3.8.2. jump

```
1 li $t0, 1
2 li $t1, 3
3 jump $t0, $t1
4 li $t3, 20
5 add $t2, $t0, $t1
```

Register	Value
0: $zero	0
1: $at	0
2: $v0	0
3: $v1	0
4: $a0	0
5: $a1	0
6: $a2	0
7: $a3	0
8: $t0	1
9: $t1	3
10: $t2	4
11: $t3	0
12: $t4	0
13: $t5	0
14: $t6	0
15: $t7	0
16: $s0	0
17: $s1	0
18: $s2	0
19: $s3	0
20: $s4	0
21: $s5	0
22: $s6	0
23: $s7	0
24: $t8	0
25: $t9	0
26: $k0	0
27: $k1	0
28: $gp	0
29: $sp	0
30: $fp	0
31: $ra	0
PC	20

3.8.3. lwi

```
1 li $t0, 30
2 sw $t0, 12($sp)
3
4 li $t1, 1
5 lw $t2, 0($sp)
6 lwi $t3, $t2, $t1, 12
```

Register	Value
0: $zero	0
1: $at	0
2: $v0	0
3: $v1	0
4: $a0	0
5: $a1	0
6: $a2	0
7: $a3	0
8: $t0	30
9: $t1	1
10: $t2	0
11: $t3	30
12: $t4	0
13: $t5	0
14: $t6	0
15: $t7	0
16: $s0	0
17: $s1	0
18: $s2	0
19: $s3	0
20: $s4	0
21: $s5	0
22: $s6	0
23: $s7	0
24: $t8	0
25: $t9	0
26: $k0	0
27: $k1	0
28: $gp	0
29: $sp	0
30: $fp	0
31: $ra	0
PC	52

4. Conclusiones

A partir de la realización del trabajo práctico se aprendió a utilizar el programa Dr. Mips, que nos permitió entender tanto en la teoría como en la práctica cómo funciona un datapath. Si bien al inicio comprender el uso de la herramienta parece complejo, luego de realizar la primera instrucción se puede apreciar el valor que tiene Dr.Mips y como facilita el entendimiento y la implementación de este tipo de acciones. También se pudo ver con claridad la diferencia entre el datapath unicycle y el pipeline.

Referencias

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

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