Trabajo Práctico 3

[66.20/86.37] Organización de Computadoras
Curso 2
Segundo cuatrimestre de 2020

Alumnos	Padrón	Correo electrónico	Slack
Gómez, Joaquín	103735	joagomez@fi.uba.ar	Joaquín Gomez
Grassano, Bruno	103855	bgrassano@fi.uba.ar	Bruno Grassano
Romero, Adrián	103371	adromero@fi.uba.ar	Adrián Romero

Índice

1. Introducción 2

2. Diseño e implementación 2
2.1. jr y jalr en uniciclo . . . 3
2.2. Pipeline . . . 5
2.2.1. j en pipeline . . . 5
2.2.2. jr en pipeline . . . 6
2.2.3. jalr en pipeline . . . 6

3. Portabilidad 7

4. Casos de prueba 7

5. Conclusiones 7

6. Referencias 8

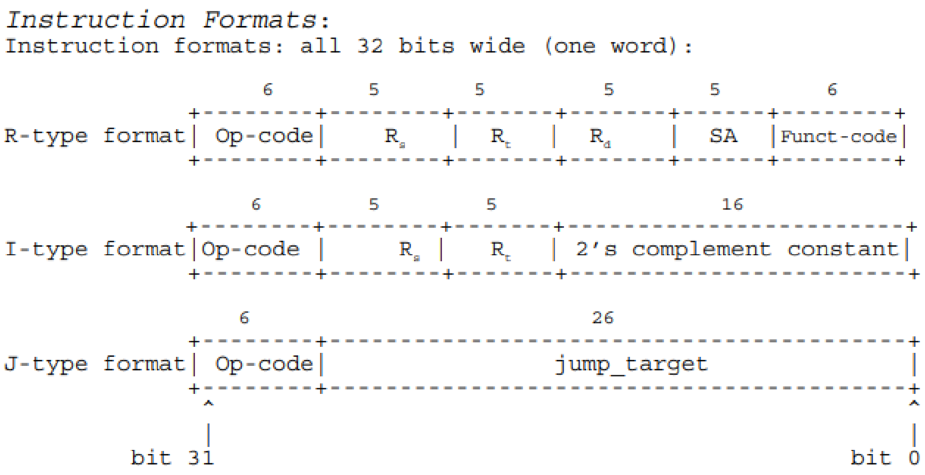
7. Apéndices 9
7.1. .set . . . 9
7.1.1. uniciclo . . . 9
7.1.2. j pipeline . . . 10
7.1.3. jr pipeline . . . 11
7.1.4. jral.set . . . 13
7.2. .cpu . . . 14
7.2.1. uniciclo . . . 14
7.2.2. j.cpu . . . 16
7.2.3. jr.cpu . . . 20
7.2.4. jalr.cpu . . . 24
7.3. Pruebas . . . 29
7.3.1. Pruebas Uniciclo . . . 29
7.3.2. Pruebas j en pipeline . . . 30
7.3.3. Pruebas jr en pipeline . . . 31
7.3.4. Pruebas jalr en pipeline . . . 33
7.4. Enunciado . . . 34

1. Introducción

El objetivo del presente trabajo es simular el datapath de la arquitectura MIPS32 agregando las implementaciones de distintas instrucciones que no vienen soportadas por defecto en el programa 'DrMips'. En particular hemos agregado las instrucciones jr y jalr en los datapath uniciclo y pipeline y la instrucción j en el datapath pipeline.

2. Diseño e implementación

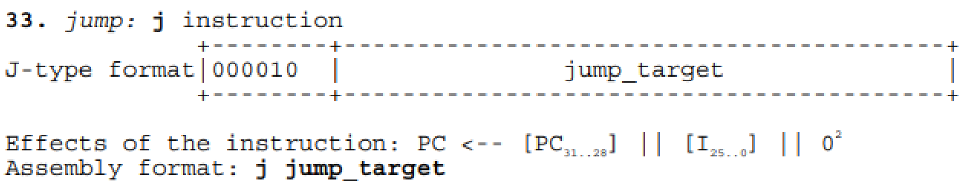
Para este trabajo se pidieron las instrucciones j, jr, y jalr. Antes de empezar con la implementación de cada una en particular, mencionamos en que consisten y mostramos los formatos de los que dispone MIPS32 para las instrucciones.



```
Instruction Formats:
Instruction formats: all 32 bits wide (one word):

                     6        5        5        5       5        6
                +--------+--------+--------+--------+-------+--------+
R-type format| Op-code|   Rs   |   Rt   |   Rd   |  SA   |Funct-code|
                +--------+--------+--------+--------+-------+--------+

                     6        5        5               16
                +--------+--------+--------+-------------------------+
I-type format|Op-code |    Rs  |   Rt   | 2's complement constant|
                +--------+--------+--------+-------------------------+

                    6                          26
                +--------+------------------------------------------+
J-type format| Op-code|               jump_target                |
                +--------+------------------------------------------+
                 ^                                                   ^
                 |                                                   |
              bit 31                                              bit 0
```

Figura 1: Formatos de las instrucciones

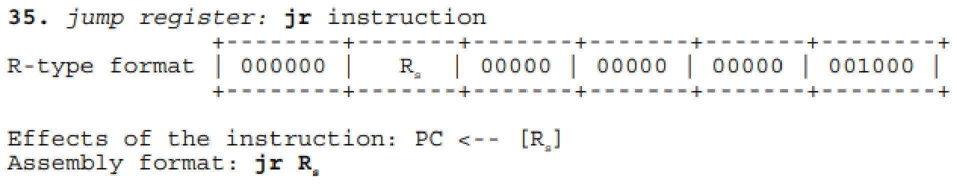
j Salta a la posición indicada. Esta es cargada en los 26 bits de la instrucción. Su uso es *j addr28*. Cabe destacar que como es múltiplo de 4 no se guardan los dos bits menos significativos.



```
33. jump: j instruction
              +--------+------------------------------------------+
J-type format|000010  |               jump_target                |
              +--------+------------------------------------------+

Effects of the instruction: PC <-- [PC31..28] || [I25..0] || 0²
Assembly format: j jump_target
```

Figura 2: Formato de la instrucción j

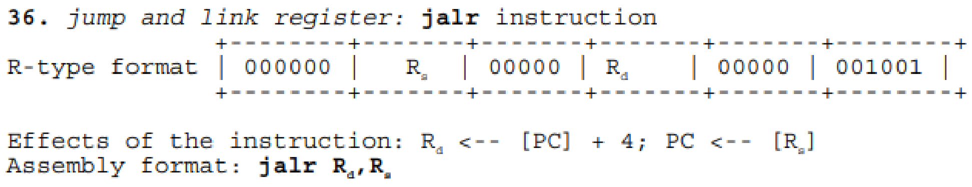
jr Salta a la posición que contiene un registro. Su uso es *jr rs*



```
35. jump register: jr instruction
                 +--------+-------+-------+-------+-------+--------+
R-type format    | 000000 |   Rs  | 00000 | 00000 | 00000 | 001000 |
                 +--------+-------+-------+-------+-------+--------+

Effects of the instruction: PC <-- [Rs]
Assembly format: jr Rs
```

Figura 3: Formato de la instrucción jr

jalr Salta a la posición que indica el registro *rs* y carga en el registro *rd* la dirección de retorno. Típicamente el registro *ra*. Su uso es *jalr rd,rs*



```
36. jump and link register: jalr instruction
                 +--------+-------+-------+-------+-------+--------+
R-type format    | 000000 |   Rs  | 00000 | Rd    | 00000 | 001001 |
                 +--------+-------+-------+-------+-------+--------+

Effects of the instruction: Rd <-- [PC] + 4; PC <-- [Rs]
Assembly format: jalr Rd,Rs
```

Figura 4: Formato de la instrucción jalr

2.1. jr y jalr en uniciclo

Dado que el datapath uniciclo es relativamente simple, hemos decidido implementar las instrucciones jalr y jr en el mismo datapath. Mostramos en la siguiente imagen como es el datapath resultante y procedemos a explicar lo que realizamos para lograr que funcionen estas instrucciones.

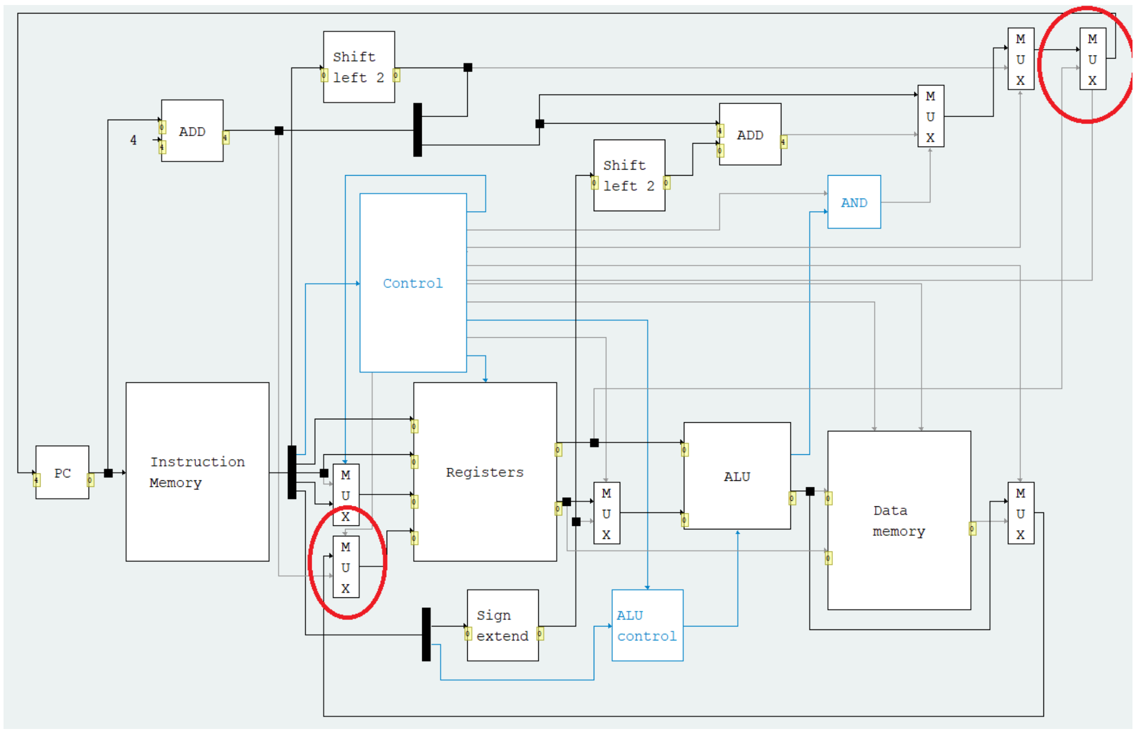


Figura 5: Circuito nuevo de uniciclo

Como se puede ver, hemos agregado dos salidas de control llamadas: ’JumpReg’ y ’JalReg’. Ambas son entradas de selección de los multiplexores que hemos agregado.

La serie de tres multiplexores en cascada permite elegir si la dirección de la instrucción siguiente proviene de: una dirección brancheada si es un branch, la dirección de salto si es un jump, y el registro de RegBank correspondiente si es un jr o jalr. Es por esto que las entradas de selección de estos multiplexores son (en orden de izquierda a derecha): ’branch’, ’jump’, "jumpReg".

Dado que ambas instrucciones jr y jalr saltan a registros, el ultimo multiplexor, mediante su entrada de selección permite realizar este salto, por eso, ambas instrucciones tendrán valor 1 en la salida de control ’jumpReg’.

Por otro lado, agregamos un multiplexor entre el instruction memory y el register bank. Este multiplexor activa su selección mediante la salida ’JalReg’ de Control. Además multiplexa entre el PC+4 y la entrada que le llegaría normalmente. De esta manera, si ’jalReg’ vale 1, escribiremos en el RegBank el registro correspondiente. Dado que en la instrucción jalr, se escribe un registro, debemos hacer que la salida de control correspondiente a ’RegWrite’ valga 1 para esta instrucción.

2.2. Pipeline

Dado que el datapath de pipeline era considerablemente mas complejo que el datapath uniciclo hemos decidido implementar las instrucciones j, jr y jalr en datapaths distintos.

2.2.1. j en pipeline

Mostraremos en la siguiente imagen como hemos implementado la instrucción j.

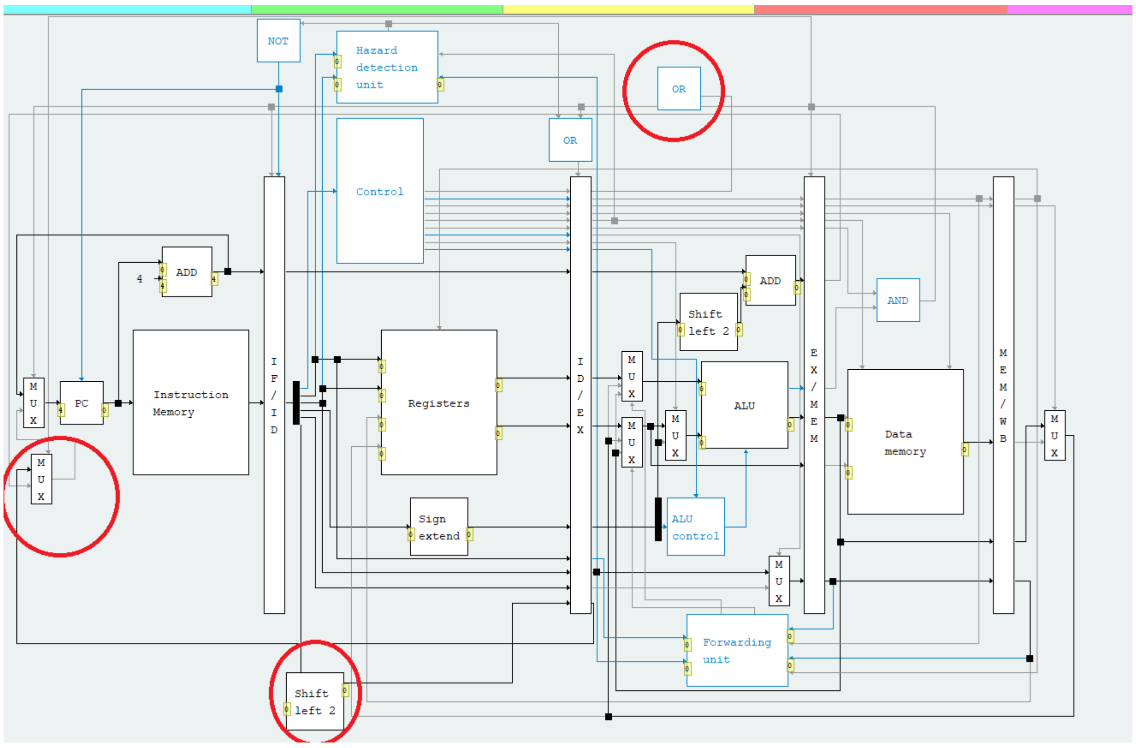


Figura 6: Circuito nuevo de pipeline con la instruccion j
hola bruno

Como se puede ver, agregamos una entrada de control llamada 'Jump' que va hacia el registro de pipeline 'ID/EX', esto nos permite llevar el valor de esta entrada de control a la siguiente etapa de la línea de ejecución de pipeline. En la etapa de ejecución del j mandamos esta señal al OR que agregamos en la parte de ejecución. De esta manera, realizaremos un flush de las dos instrucciones que se cargaron anteriormente si hay un jump o si hay un branch (además, si hay un branch también se flushea el tercer registro de pipeline).

También agregamos un shift left que toma el valor que se ingresa en la instrucción j para que este sea un múltiplo de 4 y, por lo tanto, que sea una dirección valida para después realizar el salto correspondiente.

Por último, agregamos un multiplexor para decidir que dirección se toma si se realiza un branch. Estas direcciones son las que salen del registro target de EX/MEM y la otra ,que es la que tiene el selector, es la dirección que se ingresa cuando se hace un jump que proviene de ID/EX. La salida de este multiplexor, se conecta a otro multiplexor que le envía al PC la dirección a la cual va a saltar. Esta puede ser PC+4 o la que proviene del multiplexor anterior (la cual tiene un selector que se activa con la entrada de control Jump o Branch).

2.2.2. jr en pipeline

Mostraremos en la siguiente imagen como hemos implementado la instrucción jr.

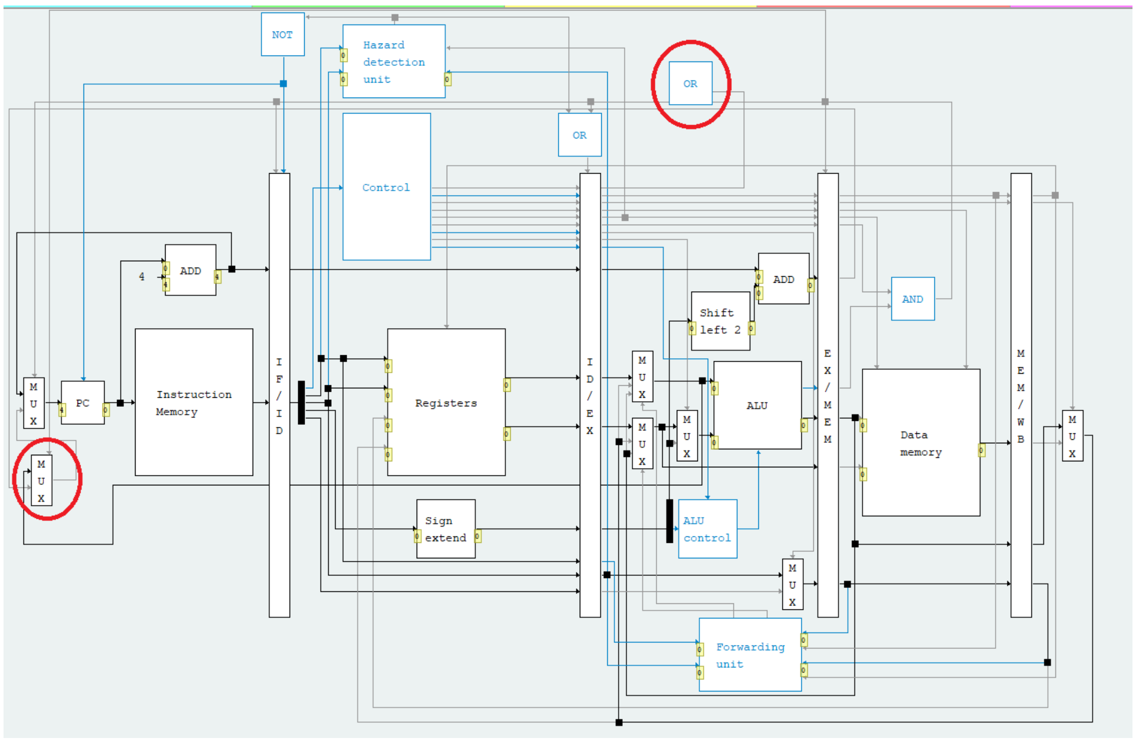


Figura 7: Circuito nuevo de pipeline con la instrucción jr

Como se puede ver, agregamos una entrada de control llamada 'JumpReg' que se dirige hacia el registro de pipeline 'ID/EX', esto logra llevar el valor de esta entrada a la siguiente etapa de la línea de ejecución de pipeline. En esta etapa mandamos esta señal al OR que agregamos en la parte de ejecución. De esta forma, conseguimos realizar un flush de las dos instrucciones que se cargaron anteriormente si hay un jump o si hay un branch (además, si hay un branch también se flushea el tercer registro de pipeline).

Por ultimo, hemos agregado un multiplexor que toma como entrada, lo mismo que recibiría la ALU, esto es por que la ALU recibe o bien el registro del RegBank que se envia de la etapa anterior, o bien un registro forwardeado de la etapa anterior, esto nos permite obtener resultados forward deseados, si es necesario. Este multiplexor tiene como selector la entrada de control "Branch", de manera que si hay que realizar un branch se toma la entrada 1 y si no hay que hacerlo se toma la entrada de la ALU, sin embargo, finalmente se lo hace pasar por otro multiplexor que logra decidir si se debe tomar como instrucción siguiente aquella en PC+4 o bien la que recibe del multiplexor anterior.

2.2.3. jalr en pipeline

Mostraremos en la siguiente imagen como hemos implementado la instrucción jalr.

Se puede ver que es muy similar al datapath pipeline anterior que implementa la instrucción jr. Esto se debe a que ambas instrucciones saltan a registros, lo que hace que podamos re-utilizar el datapath anterior.

Para implementar esta instrucción hemos tenido que agregar un multiplexor en la etapa de ejecución. Este multiplexor permite elegir el valor que queremos guardar en el registro del RegBank

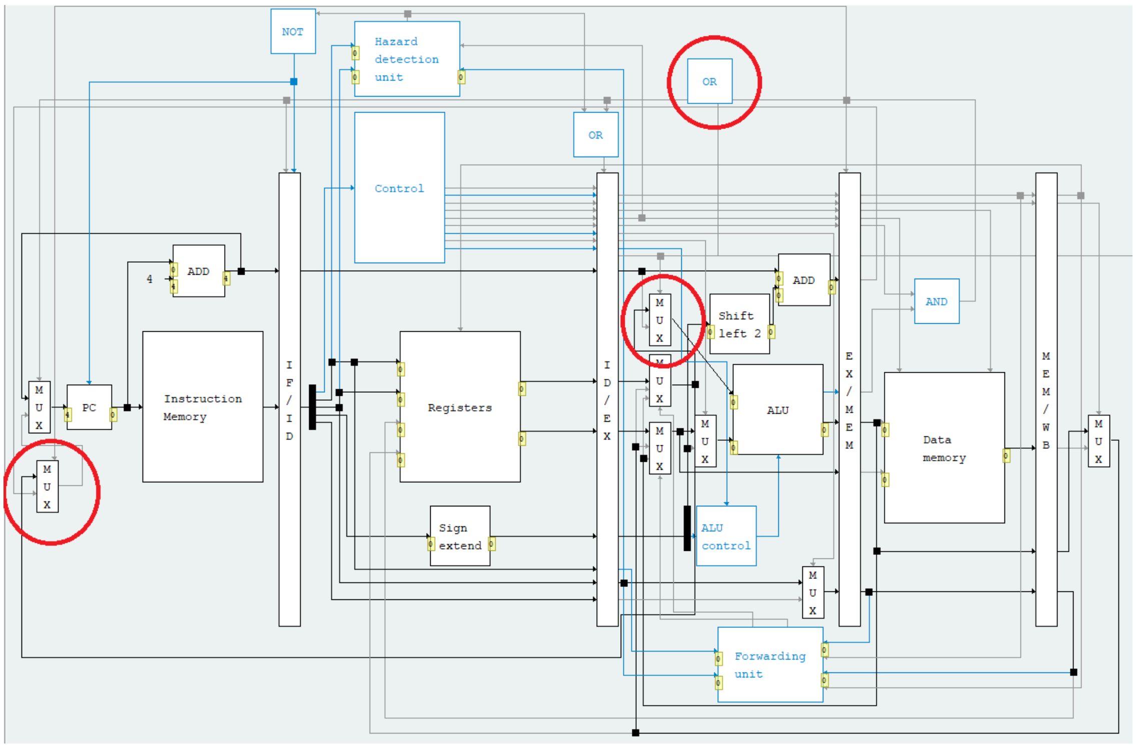


Figura 8: Circuito nuevo de pipeline con la instrucción jalr

que escribiremos, esta seleccionado por el valor de control de 'Jalr' indicando que se tomara como valor a guardar en el registro, el valor de PC+4 si Jalr vale 1.

3. Portabilidad

El trabajo practico fue diseñado utilizando los formatos 'JSON' entregados por el programa 'DrMips', por lo que es portable a los distintos sistemas que este sea compatibles, ya sean basados en sistemas Unix o Windows.

4. Casos de prueba

Para los casos de prueba se armaron distintos archivos de assembly los cuales contienen programas para probar las distintas configuraciones posibles, ya sea en uniciclo o pipeline. Para ejecutarlos, agregar cada prueba individualmente en la sección 'Code' del DrMIPS. Estos archivos se agregan en el apéndice.

5. Conclusiones

Como conclusiones del trabajo practico podemos destacar que se pudo observar el funcionamiento del camino de datos al ir ejecutando las distintas instrucciones en el programa DrMIPS. Esto aplica para ambos tipos de implementaciones, unicycle y pipeline.

También estuvo bueno observar como es que se producían los hazards en el caso del pipeline, ya que esto permitió tener mas en claro las complicaciones que estos traen y los cuidados hay que tener, junto con sus posibles soluciones para evitarlos, a pesar de las complicaciones y limitaciones que traía el DrMIPS al momento de aplicarlas.

6. Referencias

Hennessy, John L. and Patterson, David A., Computer Architecture: A Quantitative Approach, Third Edition, 2002.

https://brunonova.github.io/drmips/

7. Apéndices

7.1. .set

7.1.1. uniciclo

```
{
        "comment": "Instruction set of the reference book.",
        "types": {
                "R": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
                "I": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
                "J": [{"id": "op", "size": 6}, {"id": "target", "size": 26}],
        },
        "instructions": {
                "nop":  {"type": "R", "fields": {"op": 0, "rs": 0, "rt": 0, "rd": 0, "shamt": 0,
                "add":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "sub":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "and":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "or":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "nor":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "
                "slt":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "j":    {"type": "J", "args": ["target"], "fields": {"op": 2, "target": "#1"}, "
                "addi": {"type": "I", "args": ["reg", "reg", "int"], "fields": {"op": 8, "rs": "#
                "beq":  {"type": "I", "args": ["reg", "reg", "offset"], "fields": {"op": 4, "rs":
                "lw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 35, "rs": "#2.off
                "sw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 43, "rs": "#2.off

                "jr":                {"type": "R", "args": ["reg"], "fields": {"op": 3, "rs": "#1
                "jalr": {"type": "R", "args": ["reg","reg"], "fields": {"op": 5, "rs": "#2","rt":

        },
        "pseudo": {
                "li":   {"args": ["reg", "int"], "to": ["addi #1, $0, #2"], "desc": "$t1 = 22"},
                "la":   {"args": ["reg", "label"], "to": ["addi #1, $0, #2"], "desc": "$t1 = ADDR
                "move": {"args": ["reg", "reg"], "to": ["add #1, #2, $0"], "desc": "$t1 = $t2"},
                "subi": {"args": ["reg", "reg", "int"], "to": ["li $1, #3", "sub #1, #2, $1"], "d
                "sgt":  {"args": ["reg", "reg", "reg"], "to": ["slt #1, #3, #2"], "desc": "$t1 =
                "bge":  {"args": ["reg", "reg", "offset"], "to": ["slt $1, #1, #2", "beq $1, $0,
                "ble":  {"args": ["reg", "reg", "offset"], "to": ["sgt $1, #1, #2", "beq $1, $0,
                "b":    {"args": ["offset"], "to": ["beq $0, $0, #1"], "desc": "PC += offset * 4
                "neg":  {"args": ["reg", "reg"], "to": ["sub #1, $0, #2"], "desc": "$t1 = -$t2"},
                "not":  {"args": ["reg", "reg"], "to": ["nor #1, #2, $0"], "desc": "$t1 = ~$t2"}
        },
        "control": {
                "0": {"RegDst": 1, "RegWrite": 1, "ALUOp": 2, "ALUSrc": 0, "MemToReg": 0},
                "8": {"RegDst": 0, "RegWrite": 1, "ALUOp": 0, "ALUSrc": 1, "MemToReg": 0},
                "2": {"Jump": 1},
                "3": {"Jump": 0, "RegDst": 0, "RegWrite": 0, "ALUOp": 1, "ALUSrc": 0, "MemToReg":
                "4": {"ALUOp": 1, "ALUSrc": 0, "Branch": 1},
                "5": {"JalReg": 1, "RegWrite": 1, "JumpReg": 1, "RegDst": 1},
                "35": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 1, "MemRead": 1, "MemWri
                "43": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 0, "MemRead": 0, "MemWri
        },
        "alu": {
                "aluop_size": 2,
                "func_size": 6,
                "control_size": 4,
                "control": [
                        {"aluop": 0, "out": {"Operation": 2}},
                        {"aluop": 1, "out": {"Operation": 6}},
                        {"aluop": 1, "func": 8, "out": {"Operation": 6}},
                        {"aluop": 2, "func": 32, "out": {"Operation": 2}},
                        {"aluop": 2, "func": 34, "out": {"Operation": 6}},
                        {"aluop": 2, "func": 36, "out": {"Operation": 0}},
                        {"aluop": 2, "func": 37, "out": {"Operation": 1}},
                        {"aluop": 2, "func": 39, "out": {"Operation": 12}},
                        {"aluop": 2, "func": 42, "out": {"Operation": 7}},

                        {"aluop": 1, "func": 9, "out": {"Operation": 6}}
                ],
                "operations": {
                        "0": "and",
                        "1": "or",
                        "2": "add",
                        "6": "sub",
                        "7": "slt",
                        "12": "nor"
                }
        }
}
```

7.1.2. j pipeline

```
{
        "comment": "Instruction set of the reference book, without the jump instruction.",
        "types": {
                "R": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
                "I": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
                "J": [{"id": "op", "size": 6}, {"id": "target", "size": 26}],
        },
        "instructions": {
                "nop":  {"type": "R", "fields": {"op": 0, "rs": 0, "rt": 0, "rd": 0, "shamt": 0,
                "add":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "sub":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "and":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "nor":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "
                "or":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "slt":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "addi": {"type": "I", "args": ["reg", "reg", "int"], "fields": {"op": 8, "rs": "#
                "beq":  {"type": "I", "args": ["reg", "reg", "offset"], "fields": {"op": 4, "rs":
                "lw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 35, "rs": "#2.off
                "sw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 43, "rs": "#2.off
                "j":    {"type": "J", "args": ["target"], "fields": {"op": 7, "target": "#1"}, "d
        },
        "pseudo": {
                "li":   {"args": ["reg", "int"], "to": ["addi #1, $0, #2"], "desc": "$t1 = 22"},
                "la":   {"args": ["reg", "label"], "to": ["addi #1, $0, #2"], "desc": "$t1 = ADDR
                "move": {"args": ["reg", "reg"], "to": ["add #1, #2, $0"], "desc": "$t1 = $t2"},
		"subi": {"args": ["reg", "reg", "int"], "to": ["li $1, #3", "sub #1, #2, $1"], "d
		"sgt":  {"args": ["reg", "reg", "reg"], "to": ["slt #1, #3, #2"], "desc": "$t1 =
		"bge":  {"args": ["reg", "reg", "offset"], "to": ["slt $1, #1, #2", "beq $1, $0,
		"ble":  {"args": ["reg", "reg", "offset"], "to": ["sgt $1, #1, #2", "beq $1, $0,
		"b":    {"args": ["offset"], "to": ["beq $0, $0, #1"], "desc": "PC += offset * 4
		"neg":  {"args": ["reg", "reg"], "to": ["sub #1, $0, #2"], "desc": "$t1 = -$t2"},
		"not":  {"args": ["reg", "reg"], "to": ["nor #1, #2, $0"], "desc": "$t1 = ~$t2"}
	},
	"control": {
		"0": {"RegDst": 1, "RegWrite": 1, "ALUOp": 2, "ALUSrc": 0, "MemToReg": 0},
		"8": {"RegDst": 0, "RegWrite": 1, "ALUOp": 0, "ALUSrc": 1, "MemToReg": 0},
		"4": {"ALUOp": 1, "ALUSrc": 0, "Branch": 1},
		"35": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 1, "MemRead": 1, "MemWri
		"43": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 0, "MemRead": 0, "MemWri
		"7":  {"Jump": 1},
	},
	"alu": {
		"aluop_size": 2,
		"func_size": 6,
		"control_size": 4,
		"control": [
			{"aluop": 0, "out": {"Operation": 2}},
			{"aluop": 1, "out": {"Operation": 6}},
			{"aluop": 2, "func": 32, "out": {"Operation": 2}},
			{"aluop": 2, "func": 34, "out": {"Operation": 6}},
			{"aluop": 2, "func": 36, "out": {"Operation": 0}},
			{"aluop": 2, "func": 37, "out": {"Operation": 1}},
			{"aluop": 2, "func": 42, "out": {"Operation": 7}},
			{"aluop": 2, "func": 39, "out": {"Operation": 12}}
		],
		"operations": {
			"0": "and",
			"1": "or",
			"2": "add",
			"6": "sub",
			"7": "slt",
			"12": "nor"
		}
	}
}
```

7.1.3. jr pipeline

```
{
	"comment": "Instruction set of the reference book, without the jump instruction.",
	"types": {
		"R": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
		"I": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
	},
	"instructions": {
		"nop":  {"type": "R", "fields": {"op": 0, "rs": 0, "rt": 0, "rd": 0, "shamt": 0,
		"add":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
		"sub":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
		"and":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
```

```
                "nor":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "
                "or":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "slt":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "addi": {"type": "I", "args": ["reg", "reg", "int"], "fields": {"op": 8, "rs": "#
                "beq":  {"type": "I", "args": ["reg", "reg", "offset"], "fields": {"op": 4, "rs":
                "lw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 35, "rs": "#2.off
                "sw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 43, "rs": "#2.off

                "jr":   {"type": "R", "args":["reg"], "fields": {"op": 7, "rs":"#1", "rt":0, "rd"
        },
        "pseudo": {
                "li":   {"args": ["reg", "int"], "to": ["addi #1, $0, #2"], "desc": "$t1 = 22"},
                "la":   {"args": ["reg", "label"], "to": ["addi #1, $0, #2"], "desc": "$t1 = ADDR
                "move": {"args": ["reg", "reg"], "to": ["add #1, #2, $0"], "desc": "$t1 = $t2"},
                "subi": {"args": ["reg", "reg", "int"], "to": ["li $1, #3", "sub #1, #2, $1"], "d
                "sgt":  {"args": ["reg", "reg", "reg"], "to": ["slt #1, #3, #2"], "desc": "$t1 =
                "bge":  {"args": ["reg", "reg", "offset"], "to": ["slt $1, #1, #2", "beq $1, $0,
                "ble":  {"args": ["reg", "reg", "offset"], "to": ["sgt $1, #1, #2", "beq $1, $0,
                "b":    {"args": ["offset"], "to": ["beq $0, $0, #1"], "desc": "PC += offset * 4
                "neg":  {"args": ["reg", "reg"], "to": ["sub #1, $0, #2"], "desc": "$t1 = -$t2"},
                "not":  {"args": ["reg", "reg"], "to": ["nor #1, #2, $0"], "desc": "$t1 = ~$t2"}
        },
        "control": {
                "0": {"RegDst": 1, "RegWrite": 1, "ALUOp": 2, "ALUSrc": 0, "MemToReg": 0},
                "8": {"RegDst": 0, "RegWrite": 1, "ALUOp": 0, "ALUSrc": 1, "MemToReg": 0},
                "4": {"ALUOp": 1, "ALUSrc": 0, "Branch": 1},
                "35": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 1, "MemRead": 1, "MemWri
                "43": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 0, "MemRead": 0, "MemWri
                "7": {"JumpReg": 1}
        },
        "alu": {
                "aluop_size": 2,
                "func_size": 6,
                "control_size": 4,
                "control": [
                        {"aluop": 0, "out": {"Operation": 2}},
                        {"aluop": 1, "out": {"Operation": 6}},
                        {"aluop": 2, "func": 32, "out": {"Operation": 2}},
                        {"aluop": 2, "func": 34, "out": {"Operation": 6}},
                        {"aluop": 2, "func": 36, "out": {"Operation": 0}},
                        {"aluop": 2, "func": 37, "out": {"Operation": 1}},
                        {"aluop": 2, "func": 42, "out": {"Operation": 7}},
                        {"aluop": 2, "func": 39, "out": {"Operation": 12}}
                ],
                "operations": {
                        "0": "and",
                        "1": "or",
                        "2": "add",
                        "6": "sub",
                        "7": "slt",
                        "12": "nor"
                }
        }
}
```

7.1.4. jral.set

```
{
        "comment": "Instruction set of the reference book, without the jump instruction.",
        "types": {
                "R": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
                "I": [{"id": "op", "size": 6}, {"id": "rs", "size": 5}, {"id": "rt", "size": 5},
        },
        "instructions": {
                "nop":  {"type": "R", "fields": {"op": 0, "rs": 0, "rt": 0, "rd": 0, "shamt": 0,
                "add":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "sub":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "and":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "nor":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "
                "or":   {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "slt":  {"type": "R", "args": ["reg", "reg", "reg"], "fields": {"op": 0, "rs": "#
                "addi": {"type": "I", "args": ["reg", "reg", "int"], "fields": {"op": 8, "rs": "#
                "beq":  {"type": "I", "args": ["reg", "reg", "offset"], "fields": {"op": 4, "rs":
                "lw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 35, "rs": "#2.off
                "sw":   {"type": "I", "args": ["reg", "data"], "fields": {"op": 43, "rs": "#2.off

                "jalr": {"type": "R", "args": ["reg","reg"], "fields": {"op": 5, "rs": "#2","rt":
        },
        "pseudo": {
                "li":   {"args": ["reg", "int"], "to": ["addi #1, $0, #2"], "desc": "$t1 = 22"},
                "la":   {"args": ["reg", "label"], "to": ["addi #1, $0, #2"], "desc": "$t1 = ADDR
                "move": {"args": ["reg", "reg"], "to": ["add #1, #2, $0"], "desc": "$t1 = $t2"},
                "subi": {"args": ["reg", "reg", "int"], "to": ["li $1, #3", "sub #1, #2, $1"], "d
                "sgt":  {"args": ["reg", "reg", "reg"], "to": ["slt #1, #3, #2"], "desc": "$t1 =
                "bge":  {"args": ["reg", "reg", "offset"], "to": ["slt $1, #1, #2", "beq $1, $0,
                "ble":  {"args": ["reg", "reg", "offset"], "to": ["sgt $1, #1, #2", "beq $1, $0,
                "b":    {"args": ["offset"], "to": ["beq $0, $0, #1"], "desc": "PC += offset * 4
                "neg":  {"args": ["reg", "reg"], "to": ["sub #1, $0, #2"], "desc": "$t1 = -$t2"},
                "not":  {"args": ["reg", "reg"], "to": ["nor #1, #2, $0"], "desc": "$t1 = ~$t2"}
        },
        "control": {
                "0": {"RegDst": 1, "RegWrite": 1, "ALUOp": 2, "ALUSrc": 0, "MemToReg": 0},
                "8": {"RegDst": 0, "RegWrite": 1, "ALUOp": 0, "ALUSrc": 1, "MemToReg": 0},
                "4": {"ALUOp": 1, "ALUSrc": 0, "Branch": 1},
                "35": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 1, "MemRead": 1, "MemWri
                "43": {"ALUOp": 0, "ALUSrc": 1, "RegDst": 0, "RegWrite": 0, "MemRead": 0, "MemWri
                "5": {"JumpReg": 1, "RegWrite": 1, "RegDst": 1},
        },
        "alu": {
                "aluop_size": 2,
                "func_size": 6,
                "control_size": 4,
                "control": [
                        {"aluop": 0, "out": {"Operation": 2}},
                        {"aluop": 1, "out": {"Operation": 6}},
                        {"aluop": 2, "func": 32, "out": {"Operation": 2}},
                        {"aluop": 2, "func": 34, "out": {"Operation": 6}},
                        {"aluop": 2, "func": 36, "out": {"Operation": 0}},
                        {"aluop": 2, "func": 37, "out": {"Operation": 1}},
                        {"aluop": 2, "func": 42, "out": {"Operation": 7}},
                        {"aluop": 2, "func": 39, "out": {"Operation": 12}}
                ],
                "operations": {
                        "0": "and",
                        "1": "or",
                        "2": "add",
                        "6": "sub",
                        "7": "slt",
                        "12": "nor"
                }
        }
}
```

7.2. .cpu

7.2.1. uniciclo

```
{
        "components": {
                "PC":         {"type": "PC", "x": 40, "y": 250, "in": "NewPC", "out": "PC"},
                "PCAdder":    {"type": "Add", "latency": 50, "x": 110, "y": 58, "in1": "In1", "ir
                "Const4":     {"type": "Constant", "x": 85, "y": 73, "out": "Out", "val": 4, "siz
                "RegBank":    {"type": "RegBank", "latency": 100, "x": 250, "y": 215, "num_regs":
                "InstMem":    {"type": "InstructionMemory", "latency": 300, "x": 90, "y": 215, "i
                "ForkPC":     {"type": "Fork", "x": 80, "y": 265, "size": 32, "in": "In", "out":
                "Control":    {"type": "ControlUnit", "latency": 50, "x": 220, "y": 110, "in": "C
                "DistInst":   {"type": "Distributor", "x": 180, "y": 250, "in": {"id": "Instructi
                "MuxDst":     {"type": "Multiplexer", "latency": 15, "x": 205, "y": 260, "size":
                "ForkRt":     {"type": "Fork", "x": 200, "y": 265, "size": 5, "in": "In", "out":
                "DistImm":    {"type": "Distributor", "x": 255, "y": 340, "in": {"id": "In", "siz
                "ExtendImm":  {"type": "SignExtend", "x": 280, "y": 330, "in": {"id": "In", "size
                "MuxReg":     {"type": "Multiplexer", "latency": 15, "x": 350, "y": 270, "size":
                "ShiftJump":  {"type": "ShiftLeft", "x": 200, "y": 20, "in": {"id": "In", "size":
                "DistPC":     {"type": "Distributor", "x": 250, "y": 60, "in": {"id": "In", "size
                "ConcatJump": {"type": "Concatenator", "x": 280, "y": 40, "in1": {"id": "In1", "s
                "MuxJump":    {"type": "Multiplexer", "latency": 15, "x": 580, "y": 18, "size": 3
                "ALUControl": {"type": "ALUControl", "latency": 50, "x": 360, "y": 330, "aluop":
                "ALU":        {"type": "ALU", "latency": 100, "x": 400, "y": 237, "in1": "In1", "
                "ForkImm":    {"type": "Fork", "x": 340, "y": 292, "size": 32, "in": "In", "out":
                "ShiftImm":   {"type": "ShiftLeft", "x": 350, "y": 80, "amount": 2, "in": {"id":
                "AddBranch":  {"type": "Add", "latency": 50, "x": 420, "y": 60, "in1": "In1", "in
                "ForkBranch": {"type": "Fork", "x": 320, "y": 71, "size": 32, "in": "In", "out":
                "MuxBranch":  {"type": "Multiplexer", "latency": 15, "x": 530, "y": 50, "size": 3
                "AndBranch":  {"type": "And", "x": 480, "y": 100, "in1": "Branch", "in2": "Zero",
                "ForkMem":    {"type": "Fork", "x": 470, "y": 275, "size": 32, "in": "In", "out":
                "ForkReg":    {"type": "Fork", "x": 335, "y": 281, "size": 32, "in": "In", "out":
                "DataMem":    {"type": "DataMemory", "latency": 400, "x": 480, "y": 242, "size":
                "MuxMem":     {"type": "Multiplexer", "latency": 15, "x": 580, "y": 270, "size":

                "MuxJalr" :   {"type": "Multiplexer", "latency": 15, "x": 205, "y": 300, "size":
                "MuxJump2":   {"type": "Multiplexer", "latency": 15, "x": 620, "y": 18, "size": 3
                "ForkJump":   {"type": "Fork", "x": 350, "y": 248, "size": 32, "in": "EntradaFork
                "ForkJalr":   {"type": "Fork", "x": 175, "y": 75, "size": 32, "in": "In", "out":
        },
        "wires": [
                {"from": "PC", "out": "PC", "to": "ForkPC", "in": "In"},
                {"from": "Const4", "out": "Out", "to": "PCAdder", "in": "In2"},
                {"from": "ForkPC", "out": "Out1", "to": "PCAdder", "in": "In1", "points": [{"x":
                {"from": "ForkPC", "out": "Out2", "to": "InstMem", "in": "Address"},
                {"from": "Control", "out": "RegWrite", "to": "RegBank", "in": "RegWrite", "start"
                {"from": "InstMem", "out": "Instruction", "to": "DistInst", "in": "Instruction"},
                {"from": "DistInst", "out": "31-26", "to": "Control", "in": "Opcode", "start": {"
                {"from": "DistInst", "out": "25-21", "to": "RegBank", "in": "ReadReg1", "start":
                {"from": "DistInst", "out": "20-16", "to": "ForkRt", "in": "In", "start": {"x": 1
                {"from": "ForkRt", "out": "Out1", "to": "RegBank", "in": "ReadReg2", "points": [{
                {"from": "ForkRt", "out": "Out2", "to": "MuxDst", "in": "0", "points": [{"x": 200
                {"from": "DistInst", "out": "15-11", "to": "MuxDst", "in": "1", "start": {"x": 18
                {"from": "DistInst", "out": "15-0", "to": "DistImm", "in": "In", "start": {"x": 1
                {"from": "MuxDst", "out": "Out", "to": "RegBank", "in": "WriteReg", "end": {"x":
                {"from": "Control", "out": "RegDst", "to": "MuxDst", "in": "RegDst", "start": {"x
                {"from": "DistImm", "out": "15-0", "to": "ExtendImm", "in": "In"},
                {"from": "Control", "out": "ALUSrc", "to": "MuxReg", "in": "ALUSrc", "start": {"x
                {"from": "RegBank", "out": "ReadData2", "to": "ForkReg", "in": "In"},
                {"from": "ForkReg", "out": "Out1", "to": "MuxReg", "in": "0"},
                {"from": "ExtendImm", "out": "Out", "to": "ForkImm", "in": "In", "points": [{"x":
                {"from": "ForkImm", "out": "Out1", "to": "MuxReg", "in": "1"},
                {"from": "DistInst", "out": "25-0", "to": "ShiftJump", "in": "In", "start": {"x":
                {"from": "DistPC", "out": "31-28", "to": "ConcatJump", "in": "In1", "start": {"x"
                {"from": "ShiftJump", "out": "Out", "to": "ConcatJump", "in": "In2"},
                {"from": "DistPC", "out": "31-0", "to": "ForkBranch", "in": "In", "start": {"x":
                {"from": "ConcatJump", "out": "Out", "to": "MuxJump", "in": "1"},


                {"from": "MuxJump", "out": "Out", "to": "MuxJump2", "in": "0", "start": {"x": 580
                {"from": "MuxJump2", "out": "Out", "to": "PC", "in": "NewPC", "points": [{"x": 62
                {"from": "RegBank","out":"ReadData1","to":"ForkJump","in":"EntradaForkJump","poin
                {"from": "ForkJump","out":"Out1","to":"ALU","in":"In1","end":{"x":400,"y":248}},
                {"from": "ForkJump","out":"Out2","to":"MuxJump2","in":"1","points":[{"x": 350, "y

                {"from": "ForkJalr","out":"Out1","to":"MuxJalr","in":"1","points":[{"x": 175, "y"

                {"from": "PCAdder","out":"PC+4","to":"ForkJalr","in":"In","points":[]},

                {"from": "ForkJalr", "out": "Out2", "to": "DistPC", "in": "In"},
                {"from": "MuxJalr", "out": "Out", "to":"RegBank", "in":"WriteData", "points":[{"x

                {"from": "Control", "out": "Jump", "to": "MuxJump", "in": "Jump", "start": {"x":
                {"from": "DistImm", "out": "5-0", "to": "ALUControl", "in": "func", "points": [{"
                {"from": "Control", "out": "ALUOp", "to": "ALUControl", "in": "ALUOp", "start": {
                {"from": "MuxReg", "out": "Out", "to": "ALU", "in": "In2", "end": {"x": 400, "y":
                {"from": "ALUControl", "out": "Operation", "to": "ALU", "in": "Operation", "point
                {"from": "ForkImm", "out": "Out2", "to": "ShiftImm", "in": "In", "points": [{"x":
                {"from": "ShiftImm", "out": "Out", "to": "AddBranch", "in": "In2", "points": [{"x
                {"from": "ForkBranch", "out": "Out1", "to": "AddBranch", "in": "In1"},
                {"from": "AddBranch", "out": "Out", "to": "MuxBranch", "in": "1", "end": {"x": 53
                {"from": "ForkBranch", "out": "Out2", "to": "MuxBranch", "in": "0", "points": [{"
```

```
                {"from": "MuxBranch", "out": "Out", "to": "MuxJump", "in": "0", "points": [{"x":
                {"from": "Control", "out": "Branch", "to": "AndBranch", "in": "Branch", "start":
                {"from": "ALU", "out": "Zero", "to": "AndBranch", "in": "Zero", "start": {"x": 46
                {"from": "AndBranch", "out": "Branch", "to": "MuxBranch", "in": "Branch", "end":
                {"from": "ALU", "out": "Result", "to": "ForkMem", "in": "In", "start": {"x": 460,
                {"from": "ForkMem", "out": "Out1", "to": "DataMem", "in": "Address"},
                {"from": "ForkReg", "out": "Out2", "to": "DataMem", "in": "WriteData", "points":
                {"from": "Control", "out": "MemRead", "to": "DataMem", "in": "MemRead", "start":
                {"from": "Control", "out": "MemWrite", "to": "DataMem", "in": "MemWrite", "start"
                {"from": "Control", "out": "MemToReg", "to": "MuxMem", "in": "MemToReg", "start":
                {"from": "DataMem", "out": "ReadData", "to": "MuxMem", "in": "1"},
                {"from": "ForkMem", "out": "Out2", "to": "MuxMem", "in": "0", "points": [{"x": 47
                {"from": "MuxMem", "out": "Out", "to": "MuxJalr", "in": "0", "points": [{"x": 600

                {"from": "Control", "out": "JumpReg", "to": "MuxJump2", "in": "JumpReg", "points"
                {"from": "Control", "out": "JalReg", "to": "MuxJalr", "in": "JalReg", "points":[{
        ],
        "reg_names": ["zero", "at", "v0", "v1", "a0", "a1", "a2", "a3", "t0", "t1", "t2", "t3", "
        "instructions": "default.set"
}
```

7.2.2. j.cpu

```
{
        "components": {
                "PC":          {"type": "PC", "x": 40, "y": 250, "in": "NewPC", "out": "PC", "writ
                "ForkPC":      {"type": "Fork", "x": 80, "y": 265, "size": 32, "in": "In", "out":
                "PCAdder":     {"type": "Add", "latency": 50, "x": 110, "y": 158, "in1": "In1", "i
                "Const4":      {"type": "Constant", "x": 85, "y": 173, "out": "Out", "val": 4, "si
                "ForkPCAdder":{"type": "Fork", "x": 155, "y": 175, "size": 32, "in": "In", "out":
                "MuxPC":       {"type": "Multiplexer", "latency": 15, "x": 15, "y": 248, "size": 3
                "MuxPC2":      {"type": "Multiplexer", "latency": 15, "x": 20, "y": 300, "size": 3

                "InstMem":     {"type": "InstructionMemory", "latency": 300, "x": 90, "y": 215, "i

                "IF/ID":       {"type": "PipelineRegister", "x": 180, "y": 110, "write": "Write",

                "DistInst":    {"type": "Distributor", "x": 200, "y": 250, "in": {"id": "Instructi
                "ForkRt":      {"type": "Fork", "x": 220, "y": 265, "size": 5, "in": "In", "out":
                "RegBank":     {"type": "RegBank", "latency": 100, "x": 260, "y": 215, "num_regs":
                "Control":     {"type": "ControlUnit", "latency": 50, "x": 230, "y": 70, "in": "Op
                "ExtendImm":   {"type": "SignExtend", "x": 280, "y": 330, "in": {"id": "In", "size
                "ForkRs":      {"type": "Fork", "x": 230, "y": 235, "size": 5, "in": "In", "out":
                "HazardUnit": {"type": "HazardDetectionUnit", "latency": 50, "x": 230, "y": 10, "
                "ForkStall":   {"type": "Fork", "x": 265, "y": 5, "size": 1, "in": "In", "out": ["
                "ForkRs2":     {"type": "Fork", "x": 215, "y": 235, "size": 5, "in": "In", "out":
                "ForkRt3":     {"type": "Fork", "x": 220, "y": 255, "size": 5, "in": "In", "out":
                "NotStall":    {"type": "Not", "x": 175, "y": 2, "in": "Stall", "out": "Write"},
                "ForkWrite":   {"type": "Fork", "x": 190, "y": 50, "size": 1, "in": "In", "out": [
                "ID/EX":       {"type": "PipelineRegister", "x": 390, "y": 110, "regs": {"JumpPC":

                "ForkReg":     {"type": "Fork", "x": 445, "y": 281, "size": 32, "in": "In", "out":
                "MuxFwdA":     {"type": "Multiplexer", "latency": 15, "x": 425, "y": 230, "size":
                "MuxFwdB":     {"type": "Multiplexer", "latency": 15, "x": 425, "y": 275, "size":
```

```
        "ForkEXR2":   {"type": "Fork", "x": 421, "y": 299, "size": 32, "in": "In", "out":
        "ForkMEMR2":  {"type": "Fork", "x": 416, "y": 291, "size": 32, "in": "In", "out":
        "MuxReg":     {"type": "Multiplexer", "latency": 15, "x": 455, "y": 270, "size":
        "DistImm":    {"type": "Distributor", "x": 448, "y": 330, "in": {"id": "In", "siz
        "ALUControl": {"type": "ALUControl", "latency": 50, "x": 456, "y": 330, "aluop":
        "ALU":        {"type": "ALU", "latency": 100, "x": 480, "y": 237, "in1": "In1", "
        "MuxDst":     {"type": "Multiplexer", "latency": 15, "x": 526, "y": 370, "size":
        "ShiftImm":   {"type": "ShiftLeft", "x": 465, "y": 190, "in": {"id": "In", "size"
        "AddBranch":  {"type": "Add", "latency": 50, "x": 510, "y": 164, "in1": "In1", "i
        "ForkImm":    {"type": "Fork", "x": 450, "y": 292, "size": 32, "in": "In", "out":
        "ForkRt2":    {"type": "Fork", "x": 408, "y": 381, "size": 5, "in": "In", "out":
        "ForwardingUnit":{"type": "ForwardingUnit", "latency": 50, "x": 470, "y": 410, "e

        "EX/MEM":     {"type": "PipelineRegister", "x": 550, "y": 110, "regs": {"Result":

        "ForkMem":    {"type": "Fork", "x": 575, "y": 275, "size": 32, "in": "In", "out":
        "ForkEXR1":   {"type": "Fork", "x": 575, "y": 360, "size": 32, "in": "In", "out":
        "DataMem":    {"type": "DataMemory", "latency": 400, "x": 580, "y": 242, "size":
        "AndBranch":  {"type": "And", "x": 600, "y": 180, "in1": "Branch", "in2": "Zero",
        "ForkDst1":   {"type": "Fork", "x": 570, "y": 387, "size": 5, "in": "In", "out":
        "ForkRegWR1": {"type": "Fork", "x": 670, "y": 125, "size": 1, "in": "In", "out":
        "ForkMemRd":  {"type": "Fork", "x": 420, "y": 140, "size": 1, "in": "In", "out":
        "ForkBr1":    {"type": "Fork", "x": 555, "y": 62, "size": 1, "in": "In", "out": [
        "ForkBr2":    {"type": "Fork", "x": 397, "y": 62, "size": 1, "in": "In", "out": [
        "ForkBr3":    {"type": "Fork", "x": 185, "y": 62, "size": 1, "in": "In", "out": [
        "OrFlush":    {"type": "Or", "x": 375, "y": 70, "in1": "Stall", "in2": "Branch",
        "OrJumpFlush": {"type": "Or", "x": 450, "y": 35, "in1": "And", "in2":"Jump", "out

        "MEM/WB":     {"type": "PipelineRegister", "x": 680, "y": 110, "regs": {"Result":

        "MuxMem":     {"type": "Multiplexer", "latency": 15, "x": 715, "y": 270, "size":
        "ForkRegWR2": {"type": "Fork", "x": 710, "y": 125, "size": 1, "in": "In", "out":
        "ForkDst2":   {"type": "Fork", "x": 705, "y": 440, "size": 5, "in": "In", "out":
        "ForkMemR1":  {"type": "Fork", "x": 416, "y": 480, "size": 32, "in": "In", "out":

        "ShiftJump":  {"type": "ShiftLeft", "x": 195, "y": 450, "in": {"id": "In", "size"
    },
    "wires": [
        {"from": "PC", "out": "PC", "to": "ForkPC", "in": "In"},
        {"from": "ForkPC", "out": "Out1", "to": "InstMem", "in": "Address"},
        {"from": "ForkPC", "out": "Out2", "to": "PCAdder", "in": "In1", "points": [{"x":
        {"from": "Const4", "out": "Out", "to": "PCAdder", "in": "In2"},
        {"from": "PCAdder", "out": "PC+4", "to": "ForkPCAdder", "in": "In"},
        {"from": "ForkPCAdder", "out": "Out1", "to": "MuxPC", "in": "0", "points": [{"x":
        {"from": "ForkPCAdder", "out": "Out2", "to": "IF/ID", "in": "NewPC", "end": {"x":
        {"from": "InstMem", "out": "Instruction", "to": "IF/ID", "in": "Instruction", "en

        {"from": "IF/ID", "out": "NewPC", "to": "ID/EX", "in": "NewPC", "start": {"x": 19
        {"from": "IF/ID", "out": "Instruction", "to": "DistInst", "in": "Instruction", "s
        {"from": "DistInst", "out": "31-26", "to": "Control", "in": "Opcode", "start": {"
        {"from": "DistInst", "out": "25-21", "to": "ForkRs2", "in": "In", "start": {"x":
        {"from": "ForkRs2", "out": "Out1", "to": "ForkRs", "in": "In", "points": [{"x": 2

        {"from": "DistInst", "out": "25-0", "to":"ShiftJump", "in":"In", "start":{"x": 20
```

```
{"from": "ForkRs2", "out": "Out2", "to": "HazardUnit", "in": "IF/ID.Rs", "points"
{"from": "ForkRs", "out": "Out1", "to": "RegBank", "in": "ReadReg1"},
{"from": "ForkRs", "out": "Out2", "to": "ID/EX", "in": "Rs", "points": [{"x": 230
{"from": "DistInst", "out": "20-16", "to": "ForkRt", "in": "In", "start": {"x": 2
{"from": "ForkRt", "out": "Out1", "to": "ForkRt3", "in": "In"},
{"from": "ForkRt3", "out": "Out1", "to": "RegBank", "in": "ReadReg2"},
{"from": "ForkRt3", "out": "Out2", "to": "HazardUnit", "in": "IF/ID.Rt", "points"
{"from": "ForkRt", "out": "Out2", "to": "ID/EX", "in": "Rt", "points": [{"x": 220
{"from": "DistInst", "out": "15-0", "to": "ExtendImm", "in": "In", "start": {"x":
{"from": "DistInst", "out": "15-11", "to": "ID/EX", "in": "Rd", "start": {"x": 20
{"from": "RegBank", "out": "ReadData1", "to": "ID/EX", "in": "ReadData1", "end":
{"from": "RegBank", "out": "ReadData2", "to": "ID/EX", "in": "ReadData2", "end":
{"from": "ExtendImm", "out": "Out", "to": "ID/EX", "in": "Imm", "end": {"x": 390,
{"from": "Control", "out": "ALUOp", "to": "ID/EX", "in": "ALUOp", "start": {"x":
{"from": "Control", "out": "ALUSrc", "to": "ID/EX", "in": "ALUSrc", "start": {"x"
{"from": "Control", "out": "RegDst", "to": "ID/EX", "in": "RegDst", "start": {"x"
{"from": "Control", "out": "Branch", "to": "ID/EX", "in": "Branch", "start": {"x"
{"from": "Control", "out": "MemRead", "to": "ID/EX", "in": "MemRead", "start": {"
{"from": "Control", "out": "MemWrite", "to": "ID/EX", "in": "MemWrite", "start":
{"from": "Control", "out": "MemToReg", "to": "ID/EX", "in": "MemToReg", "start":
{"from": "Control", "out": "RegWrite", "to": "ID/EX", "in": "RegWrite", "start":
{"from": "HazardUnit", "out": "Stall", "to": "ForkStall", "in": "In"},
{"from": "ForkStall", "out": "Out1", "to": "OrFlush", "in": "Stall", "points": [{
{"from": "OrFlush", "out": "Flush", "to": "ID/EX", "in": "Flush", "start": {"x":
{"from": "ForkStall", "out": "Out2", "to": "NotStall", "in": "Stall", "end": {"x"
{"from": "NotStall", "out": "Write", "to": "ForkWrite", "in": "In", "start": {"x"
{"from": "ForkWrite", "out": "Out1", "to": "IF/ID", "in": "Write"},
{"from": "ForkWrite", "out": "Out2", "to": "PC", "in": "Write", "points": [{"x":

{"from": "ID/EX", "out": "ReadData1", "to": "MuxFwdA", "in": "0", "start": {"x":
{"from": "MuxFwdA", "out": "Out", "to": "ALU", "in": "In1", "start": {"x": 440, "
{"from": "ID/EX", "out": "ReadData2", "to": "MuxFwdB", "in": "0", "start": {"x":
{"from": "MuxFwdB", "out": "Out", "to": "ForkReg", "in": "In", "start": {"x": 440
{"from": "ForkEXR2", "out": "Out2", "to": "MuxFwdB", "in": "2"},
{"from": "ForkMEMR2", "out": "Out2", "to": "MuxFwdB", "in": "1"},
{"from": "ForkReg", "out": "Out1", "to": "MuxReg", "in": "0"},
{"from": "ForkReg", "out": "Out2", "to": "EX/MEM", "in": "ReadData2", "points": [
{"from": "MuxReg", "out": "Out", "to": "ALU", "in": "In2", "end": {"x": 480, "y":
{"from": "ID/EX", "out": "ALUSrc", "to": "MuxReg", "in": "ALUSrc", "start": {"x":
{"from": "ID/EX", "out": "Imm", "to": "DistImm", "in": "In", "start": {"x": 405,
{"from": "ALUControl", "out": "Operation", "to": "ALU", "in": "Operation", "point
{"from": "DistImm", "out": "5-0", "to": "ALUControl", "in": "func", "start": {"x"
{"from": "ID/EX", "out": "ALUOp", "to": "ALUControl", "in": "ALUOp", "start": {"x
{"from": "ALU", "out": "Zero", "to": "EX/MEM", "in": "Zero", "start": {"x": 540,
{"from": "ALU", "out": "Result", "to": "EX/MEM", "in": "Result", "start": {"x": 5
{"from": "ID/EX", "out": "Rt", "to": "ForkRt2", "in": "In", "start": {"x": 405, "
{"from": "ForkRt2", "out": "Out1", "to": "MuxDst", "in": "0"},
{"from": "ForkRt2", "out": "Out2", "to": "ForwardingUnit", "in": "ID/EX.Rt", "poi
{"from": "ForkRt2", "out": "Out3", "to": "HazardUnit", "in": "ID/EX.Rt", "points"
{"from": "ID/EX", "out": "Rd", "to": "MuxDst", "in": "1", "start": {"x": 405, "y"
{"from": "ID/EX", "out": "RegDst", "to": "MuxDst", "in": "RegDst", "start": {"x":
{"from": "MuxDst", "out": "Out", "to": "EX/MEM", "in": "RegBankDst", "end": {"x":
```

```
{"from": "ID/EX", "out": "NewPC", "to": "AddBranch", "in": "In1", "start": {"x":
{"from": "DistImm", "out": "31-0", "to": "ForkImm", "in": "In", "start": {"x": 45
{"from": "ForkImm", "out": "Out1", "to": "MuxReg", "in": "1"},
{"from": "ForkImm", "out": "Out2", "to": "ShiftImm", "in": "In", "points": [{"x":
{"from": "ShiftImm", "out": "Out", "to": "AddBranch", "in": "In2", "points": [{"x
{"from": "AddBranch", "out": "Out", "to": "EX/MEM", "in": "Target", "end": {"x":
{"from": "ID/EX", "out": "Branch", "to": "EX/MEM", "in": "Branch", "start": {"x":
{"from": "ID/EX", "out": "MemRead", "to": "ForkMemRd", "in": "In", "start": {"x":
{"from": "ForkMemRd", "out": "Out1", "to": "EX/MEM", "in": "MemRead", "end": {"x"
{"from": "ForkMemRd", "out": "Out2", "to": "HazardUnit", "in": "ID/EX.MemRead", "
{"from": "ID/EX", "out": "MemWrite", "to": "EX/MEM", "in": "MemWrite", "start": {
{"from": "ID/EX", "out": "MemToReg", "to": "EX/MEM", "in": "MemToReg", "start": {
{"from": "ID/EX", "out": "RegWrite", "to": "EX/MEM", "in": "RegWrite", "start": {
{"from": "ID/EX", "out": "Rs", "to": "ForwardingUnit", "in": "ID/EX.Rs", "start":
{"from": "ForwardingUnit", "out": "ForwardA", "to": "MuxFwdA", "in": "ForwardA",
{"from": "ForwardingUnit", "out": "ForwardB", "to": "MuxFwdB", "in": "ForwardB",

{"from": "EX/MEM", "out": "RegBankDst", "to": "ForkDst1", "in": "In", "start": {"
{"from": "ForkDst1", "out": "Out1", "to": "MEM/WB", "in": "RegBankDst", "end": {"
{"from": "ForkDst1", "out": "Out2", "to": "ForwardingUnit", "in": "EX/MEM.Rd", "p
{"from": "EX/MEM", "out": "Result", "to": "ForkMem", "in": "In", "start": {"x": 5
{"from": "ForkMem", "out": "Out1", "to": "DataMem", "in": "Address"},
{"from": "ForkMem", "out": "Out2", "to": "ForkEXR1", "in": "In"},
{"from": "ForkEXR1", "out": "Out1", "to": "MEM/WB", "in": "Result", "end": {"x":
{"from": "ForkEXR1", "out": "Out2", "to": "ForkEXR2", "in": "In", "points": [{"x"
{"from": "ForkEXR2", "out": "Out1", "to": "MuxFwdA", "in": "2", "points": [{"x":
{"from": "EX/MEM", "out": "ReadData2", "to": "DataMem", "in": "WriteData", "start
{"from": "DataMem", "out": "ReadData", "to": "MEM/WB", "in": "ReadData", "end": {
{"from": "EX/MEM", "out": "Zero", "to": "AndBranch", "in": "Zero", "start": {"x":
{"from": "EX/MEM", "out": "Branch", "to": "AndBranch", "in": "Branch", "start": {
{"from": "AndBranch", "out": "Branch", "to": "ForkBr1", "in": "In", "points": [{"
{"from": "ForkBr1", "out": "Out2", "to": "EX/MEM", "in": "Flush"},
{"from": "ForkBr2", "out": "Out1", "to": "ForkBr3", "in": "In"},
{"from": "ForkBr2", "out": "Out2", "to": "OrFlush", "in": "Branch", "end": {"x":
{"from": "ForkBr3", "out": "Out1", "to": "MuxPC", "in": "PCSrc", "points": [{"x":


{"from": "EX/MEM", "out": "MemRead", "to": "DataMem", "in": "MemRead", "start": {
{"from": "EX/MEM", "out": "MemWrite", "to": "DataMem", "in": "MemWrite", "start":
{"from": "EX/MEM", "out": "MemToReg", "to": "MEM/WB", "in": "MemToReg", "start":
{"from": "EX/MEM", "out": "RegWrite", "to": "ForkRegWR1", "in": "In", "start": {"
{"from": "ForkRegWR1", "out": "Out1", "to": "MEM/WB", "in": "RegWrite", "end": {"
{"from": "ForkRegWR1", "out": "Out2", "to": "ForwardingUnit", "in": "EX/MEM.RegWr

{"from": "MEM/WB", "out": "ReadData", "to": "MuxMem", "in": "1", "start": {"x": 6
{"from": "MEM/WB", "out": "Result", "to": "MuxMem", "in": "0", "start": {"x": 695
{"from": "MEM/WB", "out": "MemToReg", "to": "MuxMem", "in": "MemToReg", "start":
{"from": "MEM/WB", "out": "RegBankDst", "to": "ForkDst2", "in": "In", "start": {"
{"from": "ForkDst2", "out": "Out1", "to": "RegBank", "in": "WriteReg", "points":
{"from": "ForkDst2", "out": "Out2", "to": "ForwardingUnit", "in": "MEM/WB.Rd"},
{"from": "MuxMem", "out": "Out", "to": "ForkMemR1", "in": "In", "points": [{"x":
{"from": "ForkMemR1", "out": "Out1", "to": "RegBank", "in": "WriteData", "points"
{"from": "ForkMemR1", "out": "Out2", "to": "ForkMEMR2", "in": "In"},
{"from": "ForkMEMR2", "out": "Out1", "to": "MuxFwdA", "in": "1", "points": [{"x":
```

```
                {"from": "MEM/WB", "out": "RegWrite", "to": "ForkRegWR2", "in": "In", "start": {"
                {"from": "ForkRegWR2", "out": "Out1", "to": "RegBank", "in": "RegWrite", "points"
                {"from": "ForkRegWR2", "out": "Out2", "to": "ForwardingUnit", "in": "MEM/WB.RegWr

                {"from": "Control", "out": "Jump", "to": "ID/EX", "in": "Jump", "start": {"x": 29

                {"from": "ID/EX", "out": "Jump", "to": "OrJumpFlush", "in": "Jump", "start": {"x"

                {"from": "ForkBr1", "out": "Out1", "to": "OrJumpFlush", "in": "And", "end":{"x":

                {"from": "OrJumpFlush", "out": "Flush", "to": "ForkBr2", "in": "In", "start":{"x"

                {"from": "ForkBr3", "out": "Out2", "to": "IF/ID", "in": "Flush"},

                {"from": "EX/MEM", "out": "Target", "to": "MuxPC2", "in": "1", "start": {"x": 565

                {"from": "MuxPC2", "out":"Out", "to":"MuxPC", "in":"1", "points": [{"x":50, "y":
                {"from": "MuxPC", "out":"Out", "to":"PC", "in":"NewPC"},


                {"from": "ForkBr1", "out": "Out3", "to": "MuxPC2", "in": "Branch", "points": [{"x

                {"from": "ShiftJump", "out": "Out", "to":"ID/EX", "in": "JumpPC", "start": {"x":2

                {"from": "ID/EX", "out": "JumpPC", "to":"MuxPC2", "in": "0", "start": {"x":404, "


        ],
        "reg_names": ["zero", "at", "v0", "v1", "a0", "a1", "a2", "a3", "t0", "t1", "t2", "t3", "
        "instructions": "default-jump.set"
}
```

7.2.3. jr.cpu

```
{
        "components": {
                "PC":          {"type": "PC", "x": 40, "y": 250, "in": "NewPC", "out": "PC", "writ
                "ForkPC":      {"type": "Fork", "x": 80, "y": 265, "size": 32, "in": "In", "out":
                "PCAdder":     {"type": "Add", "latency": 50, "x": 110, "y": 158, "in1": "In1", "i
                "Const4":      {"type": "Constant", "x": 85, "y": 173, "out": "Out", "val": 4, "si
                "ForkPCAdder":{"type": "Fork", "x": 155, "y": 175, "size": 32, "in": "In", "out":
                "MuxPC":       {"type": "Multiplexer", "latency": 15, "x": 15, "y": 248, "size": 3
                "InstMem":     {"type": "InstructionMemory", "latency": 300, "x": 90, "y": 215, "i

                "IF/ID":       {"type": "PipelineRegister", "x": 180, "y": 110, "write": "Write",

                "DistInst":    {"type": "Distributor", "x": 200, "y": 250, "in": {"id": "Instructi
                "ForkRt":      {"type": "Fork", "x": 220, "y": 265, "size": 5, "in": "In", "out":
                "RegBank":     {"type": "RegBank", "latency": 100, "x": 260, "y": 215, "num_regs":
                "Control":     {"type": "ControlUnit", "latency": 50, "x": 230, "y": 70, "in": "Op
                "ExtendImm":   {"type": "SignExtend", "x": 280, "y": 330, "in": {"id": "In", "size
                "ForkRs":      {"type": "Fork", "x": 230, "y": 235, "size": 5, "in": "In", "out":
                "HazardUnit": {"type": "HazardDetectionUnit", "latency": 50, "x": 230, "y": 10, "
                "ForkStall":   {"type": "Fork", "x": 265, "y": 5, "size": 1, "in": "In", "out": ["
```

```
            "ForkRs2":    {"type": "Fork", "x": 215, "y": 235, "size": 5, "in": "In", "out":
            "ForkRt3":    {"type": "Fork", "x": 220, "y": 255, "size": 5, "in": "In", "out":
            "NotStall":   {"type": "Not", "x": 175, "y": 2, "in": "Stall", "out": "Write"},
            "ForkWrite":  {"type": "Fork", "x": 190, "y": 50, "size": 1, "in": "In", "out": [

            "ID/EX":      {"type": "PipelineRegister", "x": 390, "y": 110, "regs": {"ReadData

            "ForkReg":    {"type": "Fork", "x": 445, "y": 281, "size": 32, "in": "In", "out":
            "MuxFwdA":    {"type": "Multiplexer", "latency": 15, "x": 425, "y": 230, "size":
            "MuxFwdB":    {"type": "Multiplexer", "latency": 15, "x": 425, "y": 275, "size":
            "ForkEXR2":   {"type": "Fork", "x": 421, "y": 299, "size": 32, "in": "In", "out":
            "ForkMEMR2":  {"type": "Fork", "x": 416, "y": 291, "size": 32, "in": "In", "out":
            "MuxReg":     {"type": "Multiplexer", "latency": 15, "x": 455, "y": 270, "size":
            "DistImm":    {"type": "Distributor", "x": 448, "y": 330, "in": {"id": "In", "siz
            "ALUControl": {"type": "ALUControl", "latency": 50, "x": 456, "y": 330, "aluop":
            "ALU":        {"type": "ALU", "latency": 100, "x": 480, "y": 237, "in1": "In1", "
            "MuxDst":     {"type": "Multiplexer", "latency": 15, "x": 526, "y": 370, "size":
            "ShiftImm":   {"type": "ShiftLeft", "x": 465, "y": 190, "in": {"id": "In", "size"
            "AddBranch":  {"type": "Add", "latency": 50, "x": 510, "y": 164, "in1": "In1", "i
            "ForkImm":    {"type": "Fork", "x": 450, "y": 292, "size": 32, "in": "In", "out":
            "ForkRt2":    {"type": "Fork", "x": 408, "y": 381, "size": 5, "in": "In", "out":
            "ForwardingUnit":{"type": "ForwardingUnit", "latency": 50, "x": 470, "y": 410, "e

            "EX/MEM":     {"type": "PipelineRegister", "x": 550, "y": 110, "regs": {"Result":

            "ForkMem":    {"type": "Fork", "x": 575, "y": 275, "size": 32, "in": "In", "out":
            "ForkEXR1":   {"type": "Fork", "x": 575, "y": 360, "size": 32, "in": "In", "out":
            "DataMem":    {"type": "DataMemory", "latency": 400, "x": 580, "y": 242, "size":
            "AndBranch":  {"type": "And", "x": 600, "y": 180, "in1": "Branch", "in2": "Zero",
            "ForkDst1":   {"type": "Fork", "x": 570, "y": 387, "size": 5, "in": "In", "out":
            "ForkRegWR1": {"type": "Fork", "x": 670, "y": 125, "size": 1, "in": "In", "out":
            "ForkMemRd":  {"type": "Fork", "x": 420, "y": 140, "size": 1, "in": "In", "out":
            "ForkBr1":    {"type": "Fork", "x": 555, "y": 62, "size": 1, "in": "In", "out": [
            "ForkBr2":    {"type": "Fork", "x": 397, "y": 62, "size": 1, "in": "In", "out": [
            "ForkBr3":    {"type": "Fork", "x": 185, "y": 62, "size": 1, "in": "In", "out": [
            "OrFlush":    {"type": "Or", "x": 375, "y": 70, "in1": "Stall", "in2": "Branch",

            "MEM/WB":     {"type": "PipelineRegister", "x": 680, "y": 110, "regs": {"Result":

            "MuxMem":     {"type": "Multiplexer", "latency": 15, "x": 715, "y": 270, "size":
            "ForkRegWR2": {"type": "Fork", "x": 710, "y": 125, "size": 1, "in": "In", "out":
            "ForkDst2":   {"type": "Fork", "x": 705, "y": 440, "size": 5, "in": "In", "out":
            "ForkMemR1":  {"type": "Fork", "x": 416, "y": 480, "size": 32, "in": "In", "out":

            "OrJumpReg":  {"type": "Or", "x": 450, "y": 35, "in1": "AndBranch", "in2": "JumpR
            "MuxPC2":     {"type": "Multiplexer", "latency": 15, "x": 20, "y": 300, "size": 3
            "ForkJR":     {"type": "Fork", "x": 472, "y": 250, "size": 32, "in": "In", "out":
    },
    "wires": [
            {"from": "PC", "out": "PC", "to": "ForkPC", "in": "In"},
            {"from": "ForkPC", "out": "Out1", "to": "InstMem", "in": "Address"},
            {"from": "ForkPC", "out": "Out2", "to": "PCAdder", "in": "In1", "points": [{"x":
            {"from": "Const4", "out": "Out", "to": "PCAdder", "in": "In2"},
            {"from": "PCAdder", "out": "PC+4", "to": "ForkPCAdder", "in": "In"},
            {"from": "ForkPCAdder", "out": "Out1", "to": "MuxPC", "in": "0", "points": [{"x":
            {"from": "MuxPC", "out": "Out", "to": "PC", "in": "NewPC"},
            {"from": "ForkPCAdder", "out": "Out2", "to": "IF/ID", "in": "NewPC", "end": {"x":
            {"from": "InstMem", "out": "Instruction", "to": "IF/ID", "in": "Instruction", "en

            {"from": "IF/ID", "out": "NewPC", "to": "ID/EX", "in": "NewPC", "start": {"x": 19
            {"from": "IF/ID", "out": "Instruction", "to": "DistInst", "in": "Instruction", "s
            {"from": "DistInst", "out": "31-26", "to": "Control", "in": "Opcode", "start": {"
            {"from": "DistInst", "out": "25-21", "to": "ForkRs2", "in": "In", "start": {"x":
            {"from": "ForkRs2", "out": "Out1", "to": "ForkRs", "in": "In", "points": [{"x": 2
            {"from": "ForkRs2", "out": "Out2", "to": "HazardUnit", "in": "IF/ID.Rs", "points"
            {"from": "ForkRs", "out": "Out1", "to": "RegBank", "in": "ReadReg1"},
            {"from": "ForkRs", "out": "Out2", "to": "ID/EX", "in": "Rs", "points": [{"x": 230
            {"from": "DistInst", "out": "20-16", "to": "ForkRt", "in": "In", "start": {"x": 2
            {"from": "ForkRt", "out": "Out1", "to": "ForkRt3", "in": "In"},
            {"from": "ForkRt3", "out": "Out1", "to": "RegBank", "in": "ReadReg2"},
            {"from": "ForkRt3", "out": "Out2", "to": "HazardUnit", "in": "IF/ID.Rt", "points"
            {"from": "ForkRt", "out": "Out2", "to": "ID/EX", "in": "Rt", "points": [{"x": 220
            {"from": "DistInst", "out": "15-0", "to": "ExtendImm", "in": "In", "start": {"x":
            {"from": "DistInst", "out": "15-11", "to": "ID/EX", "in": "Rd", "start": {"x": 20
            {"from": "RegBank", "out": "ReadData1", "to": "ID/EX", "in": "ReadData1", "end":
            {"from": "RegBank", "out": "ReadData2", "to": "ID/EX", "in": "ReadData2", "end":
            {"from": "ExtendImm", "out": "Out", "to": "ID/EX", "in": "Imm", "end": {"x": 390,
            {"from": "Control", "out": "ALUOp", "to": "ID/EX", "in": "ALUOp", "start": {"x":
            {"from": "Control", "out": "ALUSrc", "to": "ID/EX", "in": "ALUSrc", "start": {"x"
            {"from": "Control", "out": "RegDst", "to": "ID/EX", "in": "RegDst", "start": {"x"
            {"from": "Control", "out": "Branch", "to": "ID/EX", "in": "Branch", "start": {"x"
            {"from": "Control", "out": "MemRead", "to": "ID/EX", "in": "MemRead", "start": {"
            {"from": "Control", "out": "MemWrite", "to": "ID/EX", "in": "MemWrite", "start":
            {"from": "Control", "out": "MemToReg", "to": "ID/EX", "in": "MemToReg", "start":
            {"from": "Control", "out": "RegWrite", "to": "ID/EX", "in": "RegWrite", "start":
            {"from": "HazardUnit", "out": "Stall", "to": "ForkStall", "in": "In"},
            {"from": "ForkStall", "out": "Out1", "to": "OrFlush", "in": "Stall", "points": [{
            {"from": "OrFlush", "out": "Flush", "to": "ID/EX", "in": "Flush", "start": {"x":
            {"from": "ForkStall", "out": "Out2", "to": "NotStall", "in": "Stall", "end": {"x"
            {"from": "NotStall", "out": "Write", "to": "ForkWrite", "in": "In", "start": {"x"
            {"from": "ForkWrite", "out": "Out1", "to": "IF/ID", "in": "Write"},
            {"from": "ForkWrite", "out": "Out2", "to": "PC", "in": "Write", "points": [{"x":

            {"from": "ID/EX", "out": "ReadData1", "to": "MuxFwdA", "in": "0", "start": {"x":

            {"from": "ID/EX", "out": "ReadData2", "to": "MuxFwdB", "in": "0", "start": {"x":
            {"from": "MuxFwdB", "out": "Out", "to": "ForkReg", "in": "In", "start": {"x": 440
            {"from": "ForkEXR2", "out": "Out2", "to": "MuxFwdB", "in": "2"},
            {"from": "ForkMEMR2", "out": "Out2", "to": "MuxFwdB", "in": "1"},
            {"from": "ForkReg", "out": "Out1", "to": "MuxReg", "in": "0"},
            {"from": "ForkReg", "out": "Out2", "to": "EX/MEM", "in": "ReadData2", "points": [
            {"from": "MuxReg", "out": "Out", "to": "ALU", "in": "In2", "end": {"x": 480, "y":
            {"from": "ID/EX", "out": "ALUSrc", "to": "MuxReg", "in": "ALUSrc", "start": {"x":
            {"from": "ID/EX", "out": "Imm", "to": "DistImm", "in": "In", "start": {"x": 405,
            {"from": "ALUControl", "out": "Operation", "to": "ALU", "in": "Operation", "point
            {"from": "DistImm", "out": "5-0", "to": "ALUControl", "in": "func", "start": {"x"
            {"from": "ID/EX", "out": "ALUOp", "to": "ALUControl", "in": "ALUOp", "start": {"x
            {"from": "ALU", "out": "Zero", "to": "EX/MEM", "in": "Zero", "start": {"x": 540,
            {"from": "ALU", "out": "Result", "to": "EX/MEM", "in": "Result", "start": {"x": 5
            {"from": "ID/EX", "out": "Rt", "to": "ForkRt2", "in": "In", "start": {"x": 405, "
            {"from": "ForkRt2", "out": "Out1", "to": "MuxDst", "in": "0"},
            {"from": "ForkRt2", "out": "Out2", "to": "ForwardingUnit", "in": "ID/EX.Rt", "poi
            {"from": "ForkRt2", "out": "Out3", "to": "HazardUnit", "in": "ID/EX.Rt", "points"
            {"from": "ID/EX", "out": "Rd", "to": "MuxDst", "in": "1", "start": {"x": 405, "y"
            {"from": "ID/EX", "out": "RegDst", "to": "MuxDst", "in": "RegDst", "start": {"x":
            {"from": "MuxDst", "out": "Out", "to": "EX/MEM", "in": "RegBankDst", "end": {"x":
            {"from": "ID/EX", "out": "NewPC", "to": "AddBranch", "in": "In1", "start": {"x":
            {"from": "DistImm", "out": "31-0", "to": "ForkImm", "in": "In", "start": {"x": 45
            {"from": "ForkImm", "out": "Out1", "to": "MuxReg", "in": "1"},
            {"from": "ForkImm", "out": "Out2", "to": "ShiftImm", "in": "In", "points": [{"x":
            {"from": "ShiftImm", "out": "Out", "to": "AddBranch", "in": "In2", "points": [{"x
            {"from": "AddBranch", "out": "Out", "to": "EX/MEM", "in": "Target", "end": {"x":
            {"from": "ID/EX", "out": "Branch", "to": "EX/MEM", "in": "Branch", "start": {"x":
            {"from": "ID/EX", "out": "MemRead", "to": "ForkMemRd", "in": "In", "start": {"x":
            {"from": "ForkMemRd", "out": "Out1", "to": "EX/MEM", "in": "MemRead", "end": {"x"
            {"from": "ForkMemRd", "out": "Out2", "to": "HazardUnit", "in": "ID/EX.MemRead", "
            {"from": "ID/EX", "out": "MemWrite", "to": "EX/MEM", "in": "MemWrite", "start": {
            {"from": "ID/EX", "out": "MemToReg", "to": "EX/MEM", "in": "MemToReg", "start": {
            {"from": "ID/EX", "out": "RegWrite", "to": "EX/MEM", "in": "RegWrite", "start": {
            {"from": "ID/EX", "out": "Rs", "to": "ForwardingUnit", "in": "ID/EX.Rs", "start":
            {"from": "ForwardingUnit", "out": "ForwardA", "to": "MuxFwdA", "in": "ForwardA",
            {"from": "ForwardingUnit", "out": "ForwardB", "to": "MuxFwdB", "in": "ForwardB",

            {"from": "EX/MEM", "out": "RegBankDst", "to": "ForkDst1", "in": "In", "start": {"
            {"from": "ForkDst1", "out": "Out1", "to": "MEM/WB", "in": "RegBankDst", "end": {"
            {"from": "ForkDst1", "out": "Out2", "to": "ForwardingUnit", "in": "EX/MEM.Rd", "p
            {"from": "EX/MEM", "out": "Result", "to": "ForkMem", "in": "In", "start": {"x": 5
            {"from": "ForkMem", "out": "Out1", "to": "DataMem", "in": "Address"},
            {"from": "ForkMem", "out": "Out2", "to": "ForkEXR1", "in": "In"},
            {"from": "ForkEXR1", "out": "Out1", "to": "MEM/WB", "in": "Result", "end": {"x":
            {"from": "ForkEXR1", "out": "Out2", "to": "ForkEXR2", "in": "In", "points": [{"x"
            {"from": "ForkEXR2", "out": "Out1", "to": "MuxFwdA", "in": "2", "points": [{"x":
            {"from": "EX/MEM", "out": "ReadData2", "to": "DataMem", "in": "WriteData", "start
            {"from": "DataMem", "out": "ReadData", "to": "MEM/WB", "in": "ReadData", "end": {
            {"from": "EX/MEM", "out": "Zero", "to": "AndBranch", "in": "Zero", "start": {"x":
            {"from": "EX/MEM", "out": "Branch", "to": "AndBranch", "in": "Branch", "start": {
            {"from": "AndBranch", "out": "Branch", "to": "ForkBr1", "in": "In", "points": [{"
            {"from": "ForkBr1", "out": "Out2", "to": "EX/MEM", "in": "Flush"},
            {"from": "ForkBr2", "out": "Out1", "to": "ForkBr3", "in": "In"},
            {"from": "ForkBr2", "out": "Out2", "to": "OrFlush", "in": "Branch", "end": {"x":
            {"from": "ForkBr3", "out": "Out1", "to": "MuxPC", "in": "PCSrc", "points": [{"x":
            {"from": "ForkBr3", "out": "Out2", "to": "IF/ID", "in": "Flush"},
            {"from": "EX/MEM", "out": "MemRead", "to": "DataMem", "in": "MemRead", "start": {
            {"from": "EX/MEM", "out": "MemWrite", "to": "DataMem", "in": "MemWrite", "start":
            {"from": "EX/MEM", "out": "MemToReg", "to": "MEM/WB", "in": "MemToReg", "start":
            {"from": "EX/MEM", "out": "RegWrite", "to": "ForkRegWR1", "in": "In", "start": {"
            {"from": "ForkRegWR1", "out": "Out1", "to": "MEM/WB", "in": "RegWrite", "end": {"
            {"from": "ForkRegWR1", "out": "Out2", "to": "ForwardingUnit", "in": "EX/MEM.RegWr

            {"from": "MEM/WB", "out": "ReadData", "to": "MuxMem", "in": "1", "start": {"x": 6
            {"from": "MEM/WB", "out": "Result", "to": "MuxMem", "in": "0", "start": {"x": 695
            {"from": "MEM/WB", "out": "MemToReg", "to": "MuxMem", "in": "MemToReg", "start":
```

```
                {"from": "MEM/WB", "out": "RegBankDst", "to": "ForkDst2", "in": "In", "start": {"
                {"from": "ForkDst2", "out": "Out1", "to": "RegBank", "in": "WriteReg", "points":
                {"from": "ForkDst2", "out": "Out2", "to": "ForwardingUnit", "in": "MEM/WB.Rd"},
                {"from": "MuxMem", "out": "Out", "to": "ForkMemR1", "in": "In", "points": [{"x":
                {"from": "ForkMemR1", "out": "Out1", "to": "RegBank", "in": "WriteData", "points"
                {"from": "ForkMemR1", "out": "Out2", "to": "ForkMEMR2", "in": "In"},
                {"from": "ForkMEMR2", "out": "Out1", "to": "MuxFwdA", "in": "1", "points": [{"x":
                {"from": "MEM/WB", "out": "RegWrite", "to": "ForkRegWR2", "in": "In", "start": {"
                {"from": "ForkRegWR2", "out": "Out1", "to": "RegBank", "in": "RegWrite", "points"
                {"from": "ForkRegWR2", "out": "Out2", "to": "ForwardingUnit", "in": "MEM/WB.RegWr

                {"from": "Control", "out":"JumpReg", "to":"ID/EX", "in":"JumpReg", "start": {"x":
                {"from": "ID/EX", "out":"JumpReg", "to":"OrJumpReg", "in":"JumpRegID/EX", "start"

                {"from": "ForkBr1", "out": "Out1", "to": "OrJumpReg", "in": "AndBranch", "end":{"

                {"from": "OrJumpReg", "out": "Out", "to": "ForkBr2", "in":"In", "start":{"x": 455
                {"from": "MuxFwdA", "out": "Out", "to": "ForkJR", "in": "In", "start": {"x": 440,
                {"from": "ForkJR", "out": "Out1", "to": "ALU", "in": "In1", "end": {"x": 480, "y"

                {"from": "ForkJR", "out": "Out2", "to": "MuxPC2", "in":"0", "points":[{"x": 472,
                {"from": "ForkBr1", "out": "Out3", "to": "MuxPC2", "in":"Branch", "points": [{"x"
                {"from": "EX/MEM", "out": "Target", "to": "MuxPC2", "in": "1", "start": {"x": 565
                {"from": "MuxPC2", "out":"Out", "to":"MuxPC", "in":"1", "points": [{"x":50, "y":

        ],
        "reg_names": ["zero", "at", "v0", "v1", "a0", "a1", "a2", "a3", "t0", "t1", "t2", "t3", "
        "instructions": "jr_pipeline-set.set"
}
```

7.2.4. jalr.cpu

```
{
        "components": {
                "PC":         {"type": "PC", "x": 40, "y": 250, "in": "NewPC", "out": "PC", "writ
                "ForkPC":     {"type": "Fork", "x": 80, "y": 265, "size": 32, "in": "In", "out":
                "PCAdder":    {"type": "Add", "latency": 50, "x": 110, "y": 158, "in1": "In1", "i
                "Const4":     {"type": "Constant", "x": 85, "y": 173, "out": "Out", "val": 4, "si
                "ForkPCAdder":{"type": "Fork", "x": 155, "y": 175, "size": 32, "in": "In", "out":
                "MuxPC":      {"type": "Multiplexer", "latency": 15, "x": 15, "y": 248, "size": 3
                "InstMem":    {"type": "InstructionMemory", "latency": 300, "x": 90, "y": 215, "i

                "MuxFwdA/PC": {"type": "Multiplexer", "latency": 15, "x": 425, "y": 190, "size":


                "IF/ID":      {"type": "PipelineRegister", "x": 180, "y": 110, "write": "Write",

                "DistInst":   {"type": "Distributor", "x": 200, "y": 250, "in": {"id": "Instructi
                "ForkRt":     {"type": "Fork", "x": 220, "y": 265, "size": 5, "in": "In", "out":
                "RegBank":    {"type": "RegBank", "latency": 100, "x": 260, "y": 215, "num_regs":
                "Control":    {"type": "ControlUnit", "latency": 50, "x": 230, "y": 70, "in": "Op
                "ExtendImm":  {"type": "SignExtend", "x": 280, "y": 330, "in": {"id": "In", "size
                "ForkRs":     {"type": "Fork", "x": 230, "y": 235, "size": 5, "in": "In", "out":
                "HazardUnit": {"type": "HazardDetectionUnit", "latency": 50, "x": 230, "y": 10, "
```

```
"ForkStall":  {"type": "Fork", "x": 265, "y": 5, "size": 1, "in": "In", "out": ["
"ForkRs2":    {"type": "Fork", "x": 215, "y": 235, "size": 5, "in": "In", "out":
"ForkRt3":    {"type": "Fork", "x": 220, "y": 255, "size": 5, "in": "In", "out":
"NotStall":   {"type": "Not", "x": 175, "y": 2, "in": "Stall", "out": "Write"},
"ForkWrite":  {"type": "Fork", "x": 190, "y": 50, "size": 1, "in": "In", "out": [



"ID/EX":      {"type": "PipelineRegister", "x": 390, "y": 110, "regs": {"ReadData

"ForkReg":    {"type": "Fork", "x": 445, "y": 281, "size": 32, "in": "In", "out":
"MuxFwdA":    {"type": "Multiplexer", "latency": 15, "x": 425, "y": 230, "size":
"MuxFwdB":    {"type": "Multiplexer", "latency": 15, "x": 425, "y": 275, "size":
"ForkEXR2":   {"type": "Fork", "x": 421, "y": 299, "size": 32, "in": "In", "out":
"ForkMEMR2":  {"type": "Fork", "x": 416, "y": 291, "size": 32, "in": "In", "out":
"MuxReg":     {"type": "Multiplexer", "latency": 15, "x": 455, "y": 270, "size":
"DistImm":    {"type": "Distributor", "x": 448, "y": 330, "in": {"id": "In", "siz
"ALUControl": {"type": "ALUControl", "latency": 50, "x": 456, "y": 330, "aluop":
"ALU":        {"type": "ALU", "latency": 100, "x": 480, "y": 237, "in1": "In1", "
"MuxDst":     {"type": "Multiplexer", "latency": 15, "x": 526, "y": 370, "size":
"ShiftImm":   {"type": "ShiftLeft", "x": 465, "y": 190, "in": {"id": "In", "size"
"AddBranch":  {"type": "Add", "latency": 50, "x": 510, "y": 164, "in1": "In1", "i
"ForkImm":    {"type": "Fork", "x": 450, "y": 292, "size": 32, "in": "In", "out":
"ForkRt2":    {"type": "Fork", "x": 408, "y": 381, "size": 5, "in": "In", "out":
"ForwardingUnit":{"type": "ForwardingUnit", "latency": 50, "x": 470, "y": 410, "e

"EX/MEM":     {"type": "PipelineRegister", "x": 550, "y": 110, "regs": {"Result":

"ForkMem":    {"type": "Fork", "x": 575, "y": 275, "size": 32, "in": "In", "out":
"ForkEXR1":   {"type": "Fork", "x": 575, "y": 360, "size": 32, "in": "In", "out":
"DataMem":    {"type": "DataMemory", "latency": 400, "x": 580, "y": 242, "size":
"AndBranch":  {"type": "And", "x": 600, "y": 180, "in1": "Branch", "in2": "Zero",
"ForkDst1":   {"type": "Fork", "x": 570, "y": 387, "size": 5, "in": "In", "out":
"ForkRegWR1": {"type": "Fork", "x": 670, "y": 125, "size": 1, "in": "In", "out":
"ForkMemRd":  {"type": "Fork", "x": 420, "y": 140, "size": 1, "in": "In", "out":
"ForkBr1":    {"type": "Fork", "x": 555, "y": 62, "size": 1, "in": "In", "out": [
"ForkBr2":    {"type": "Fork", "x": 397, "y": 62, "size": 1, "in": "In", "out": [
"ForkBr3":    {"type": "Fork", "x": 185, "y": 62, "size": 1, "in": "In", "out": [
"OrFlush":    {"type": "Or", "x": 375, "y": 70, "in1": "Stall", "in2": "Branch",

"MEM/WB":     {"type": "PipelineRegister", "x": 680, "y": 110, "regs": {"Result":

"MuxMem":     {"type": "Multiplexer", "latency": 15, "x": 715, "y": 270, "size":
"ForkRegWR2": {"type": "Fork", "x": 710, "y": 125, "size": 1, "in": "In", "out":
"ForkDst2":   {"type": "Fork", "x": 705, "y": 440, "size": 5, "in": "In", "out":
"ForkMemR1":  {"type": "Fork", "x": 416, "y": 480, "size": 32, "in": "In", "out":


"OrJumpReg":  {"type": "Or", "x": 450, "y": 35, "in1": "AndBranch", "in2": "JumpR
"MuxPC2":     {"type": "Multiplexer", "latency": 15, "x": 20, "y": 300, "size": 3
"ForkJR":     {"type": "Fork", "x": 455, "y": 250, "size": 32, "in": "In", "out":

"ForkAdder":  {"type": "Fork", "x": 420, "y": 175, "size": 32, "in": "In", "out":
"ForkSelect":  {"type": "Fork", "x": 432, "y": 165, "size": 1, "in": "In", "out":
        },
        "wires": [
                {"from": "PC", "out": "PC", "to": "ForkPC", "in": "In"},
                {"from": "ForkPC", "out": "Out1", "to": "InstMem", "in": "Address"},
                {"from": "ForkPC", "out": "Out2", "to": "PCAdder", "in": "In1", "points": [{"x":
                {"from": "Const4", "out": "Out", "to": "PCAdder", "in": "In2"},
                {"from": "PCAdder", "out": "PC+4", "to": "ForkPCAdder", "in": "In"},
                {"from": "ForkPCAdder", "out": "Out1", "to": "MuxPC", "in": "0", "points": [{"x":
                {"from": "MuxPC", "out": "Out", "to": "PC", "in": "NewPC"},
                {"from": "ForkPCAdder", "out": "Out2", "to": "IF/ID", "in": "NewPC", "end": {"x":
                {"from": "InstMem", "out": "Instruction", "to": "IF/ID", "in": "Instruction", "en

                {"from": "IF/ID", "out": "NewPC", "to": "ID/EX", "in": "NewPC", "start": {"x": 19
                {"from": "IF/ID", "out": "Instruction", "to": "DistInst", "in": "Instruction", "s
                {"from": "DistInst", "out": "31-26", "to": "Control", "in": "Opcode", "start": {"
                {"from": "DistInst", "out": "25-21", "to": "ForkRs2", "in": "In", "start": {"x":
                {"from": "ForkRs2", "out": "Out1", "to": "ForkRs", "in": "In", "points": [{"x": 2
                {"from": "ForkRs2", "out": "Out2", "to": "HazardUnit", "in": "IF/ID.Rs", "points"
                {"from": "ForkRs", "out": "Out1", "to": "RegBank", "in": "ReadReg1"},
                {"from": "ForkRs", "out": "Out2", "to": "ID/EX", "in": "Rs", "points": [{"x": 230
                {"from": "DistInst", "out": "20-16", "to": "ForkRt", "in": "In", "start": {"x": 2
                {"from": "ForkRt", "out": "Out1", "to": "ForkRt3", "in": "In"},
                {"from": "ForkRt3", "out": "Out1", "to": "RegBank", "in": "ReadReg2"},
                {"from": "ForkRt3", "out": "Out2", "to": "HazardUnit", "in": "IF/ID.Rt", "points"
                {"from": "ForkRt", "out": "Out2", "to": "ID/EX", "in": "Rt", "points": [{"x": 220
                {"from": "DistInst", "out": "15-0", "to": "ExtendImm", "in": "In", "start": {"x":
                {"from": "DistInst", "out": "15-11", "to": "ID/EX", "in": "Rd", "start": {"x": 20
                {"from": "RegBank", "out": "ReadData1", "to": "ID/EX", "in": "ReadData1", "end":
                {"from": "RegBank", "out": "ReadData2", "to": "ID/EX", "in": "ReadData2", "end":
                {"from": "ExtendImm", "out": "Out", "to": "ID/EX", "in": "Imm", "end": {"x": 390,
                {"from": "Control", "out": "ALUOp", "to": "ID/EX", "in": "ALUOp", "start": {"x":
                {"from": "Control", "out": "ALUSrc", "to": "ID/EX", "in": "ALUSrc", "start": {"x"
                {"from": "Control", "out": "RegDst", "to": "ID/EX", "in": "RegDst", "start": {"x"
                {"from": "Control", "out": "Branch", "to": "ID/EX", "in": "Branch", "start": {"x"
                {"from": "Control", "out": "MemRead", "to": "ID/EX", "in": "MemRead", "start": {"
                {"from": "Control", "out": "MemWrite", "to": "ID/EX", "in": "MemWrite", "start":
                {"from": "Control", "out": "MemToReg", "to": "ID/EX", "in": "MemToReg", "start":
                {"from": "Control", "out": "RegWrite", "to": "ID/EX", "in": "RegWrite", "start":
                {"from": "HazardUnit", "out": "Stall", "to": "ForkStall", "in": "In"},
                {"from": "ForkStall", "out": "Out1", "to": "OrFlush", "in": "Stall", "points": [{
                {"from": "OrFlush", "out": "Flush", "to": "ID/EX", "in": "Flush", "start": {"x":
                {"from": "ForkStall", "out": "Out2", "to": "NotStall", "in": "Stall", "end": {"x"
                {"from": "NotStall", "out": "Write", "to": "ForkWrite", "in": "In", "start": {"x"
                {"from": "ForkWrite", "out": "Out1", "to": "IF/ID", "in": "Write"},
                {"from": "ForkWrite", "out": "Out2", "to": "PC", "in": "Write", "points": [{"x":

                {"from": "ID/EX", "out": "ReadData1", "to": "MuxFwdA", "in": "0", "start": {"x":

                {"from": "ID/EX", "out": "ReadData2", "to": "MuxFwdB", "in": "0", "start": {"x":
                {"from": "MuxFwdB", "out": "Out", "to": "ForkReg", "in": "In", "start": {"x": 440
                {"from": "ForkEXR2", "out": "Out2", "to": "MuxFwdB", "in": "2"},
                {"from": "ForkMEMR2", "out": "Out2", "to": "MuxFwdB", "in": "1"},
                {"from": "ForkReg", "out": "Out1", "to": "MuxReg", "in": "0"},
                {"from": "ForkReg", "out": "Out2", "to": "EX/MEM", "in": "ReadData2", "points": [
                {"from": "MuxReg", "out": "Out", "to": "ALU", "in": "In2", "end": {"x": 480, "y":
                {"from": "ID/EX", "out": "ALUSrc", "to": "MuxReg", "in": "ALUSrc", "start": {"x":
                {"from": "ID/EX", "out": "Imm", "to": "DistImm", "in": "In", "start": {"x": 405,
                {"from": "ALUControl", "out": "Operation", "to": "ALU", "in": "Operation", "point
                {"from": "DistImm", "out": "5-0", "to": "ALUControl", "in": "func", "start": {"x"
                {"from": "ID/EX", "out": "ALUOp", "to": "ALUControl", "in": "ALUOp", "start": {"x
                {"from": "ALU", "out": "Zero", "to": "EX/MEM", "in": "Zero", "start": {"x": 540,
                {"from": "ALU", "out": "Result", "to": "EX/MEM", "in": "Result", "start": {"x": 5
                {"from": "ID/EX", "out": "Rt", "to": "ForkRt2", "in": "In", "start": {"x": 405, "
                {"from": "ForkRt2", "out": "Out1", "to": "MuxDst", "in": "0"},
                {"from": "ForkRt2", "out": "Out2", "to": "ForwardingUnit", "in": "ID/EX.Rt", "poi
                {"from": "ForkRt2", "out": "Out3", "to": "HazardUnit", "in": "ID/EX.Rt", "points"
                {"from": "ID/EX", "out": "Rd", "to": "MuxDst", "in": "1", "start": {"x": 405, "y"
                {"from": "ID/EX", "out": "RegDst", "to": "MuxDst", "in": "RegDst", "start": {"x":
                {"from": "MuxDst", "out": "Out", "to": "EX/MEM", "in": "RegBankDst", "end": {"x":

                {"from": "ID/EX", "out": "NewPC", "to": "ForkAdder", "in": "In", "start": {"x": 4
                {"from": "ForkAdder", "out": "Out1", "to": "MuxFwdA/PC", "in": "1", "points": [{"
                {"from": "ForkAdder", "out": "Out2", "to": "AddBranch", "in": "In1"},
                {"from": "MuxFwdA/PC", "out":"Out", "to": "ALU", "in":"In1"},

                {"from": "DistImm", "out": "31-0", "to": "ForkImm", "in": "In", "start": {"x": 45
                {"from": "ForkImm", "out": "Out1", "to": "MuxReg", "in": "1"},
                {"from": "ForkImm", "out": "Out2", "to": "ShiftImm", "in": "In", "points": [{"x":
                {"from": "ShiftImm", "out": "Out", "to": "AddBranch", "in": "In2", "points": [{"x

                {"from": "AddBranch", "out": "Out", "to": "EX/MEM", "in": "Target", "end": {"x":

                {"from": "ID/EX", "out": "Branch", "to": "EX/MEM", "in": "Branch", "start": {"x":
                {"from": "ID/EX", "out": "MemRead", "to": "ForkMemRd", "in": "In", "start": {"x":
                {"from": "ForkMemRd", "out": "Out1", "to": "EX/MEM", "in": "MemRead", "end": {"x"
                {"from": "ForkMemRd", "out": "Out2", "to": "HazardUnit", "in": "ID/EX.MemRead", "
                {"from": "ID/EX", "out": "MemWrite", "to": "EX/MEM", "in": "MemWrite", "start": {
                {"from": "ID/EX", "out": "MemToReg", "to": "EX/MEM", "in": "MemToReg", "start": {
                {"from": "ID/EX", "out": "RegWrite", "to": "EX/MEM", "in": "RegWrite", "start": {
                {"from": "ID/EX", "out": "Rs", "to": "ForwardingUnit", "in": "ID/EX.Rs", "start":
                {"from": "ForwardingUnit", "out": "ForwardA", "to": "MuxFwdA", "in": "ForwardA",
                {"from": "ForwardingUnit", "out": "ForwardB", "to": "MuxFwdB", "in": "ForwardB",

                {"from": "EX/MEM", "out": "RegBankDst", "to": "ForkDst1", "in": "In", "start": {"
                {"from": "ForkDst1", "out": "Out1", "to": "MEM/WB", "in": "RegBankDst", "end": {"
                {"from": "ForkDst1", "out": "Out2", "to": "ForwardingUnit", "in": "EX/MEM.Rd", "p
                {"from": "EX/MEM", "out": "Result", "to": "ForkMem", "in": "In", "start": {"x": 5
                {"from": "ForkMem", "out": "Out1", "to": "DataMem", "in": "Address"},
                {"from": "ForkMem", "out": "Out2", "to": "ForkEXR1", "in": "In"},
                {"from": "ForkEXR1", "out": "Out1", "to": "MEM/WB", "in": "Result", "end": {"x":
                {"from": "ForkEXR1", "out": "Out2", "to": "ForkEXR2", "in": "In", "points": [{"x"
                {"from": "ForkEXR2", "out": "Out1", "to": "MuxFwdA", "in": "2", "points": [{"x":
                {"from": "EX/MEM", "out": "ReadData2", "to": "DataMem", "in": "WriteData", "start
                {"from": "DataMem", "out": "ReadData", "to": "MEM/WB", "in": "ReadData", "end": {
                {"from": "EX/MEM", "out": "Zero", "to": "AndBranch", "in": "Zero", "start": {"x":
                {"from": "EX/MEM", "out": "Branch", "to": "AndBranch", "in": "Branch", "start": {
                {"from": "AndBranch", "out": "Branch", "to": "ForkBr1", "in": "In", "points": [{"
```

```
            {"from": "ForkBr1", "out": "Out2", "to": "EX/MEM", "in": "Flush"},
            {"from": "ForkBr2", "out": "Out1", "to": "ForkBr3", "in": "In"},
            {"from": "ForkBr2", "out": "Out2", "to": "OrFlush", "in": "Branch", "end": {"x":
            {"from": "ForkBr3", "out": "Out1", "to": "MuxPC", "in": "PCSrc", "points": [{"x":
            {"from": "ForkBr3", "out": "Out2", "to": "IF/ID", "in": "Flush"},
            {"from": "EX/MEM", "out": "MemRead", "to": "DataMem", "in": "MemRead", "start": {
            {"from": "EX/MEM", "out": "MemWrite", "to": "DataMem", "in": "MemWrite", "start":
            {"from": "EX/MEM", "out": "MemToReg", "to": "MEM/WB", "in": "MemToReg", "start":
            {"from": "EX/MEM", "out": "RegWrite", "to": "ForkRegWR1", "in": "In", "start": {"
            {"from": "ForkRegWR1", "out": "Out1", "to": "MEM/WB", "in": "RegWrite", "end": {"
            {"from": "ForkRegWR1", "out": "Out2", "to": "ForwardingUnit", "in": "EX/MEM.RegWr

            {"from": "MEM/WB", "out": "ReadData", "to": "MuxMem", "in": "1", "start": {"x": 6
            {"from": "MEM/WB", "out": "Result", "to": "MuxMem", "in": "0", "start": {"x": 695
            {"from": "MEM/WB", "out": "MemToReg", "to": "MuxMem", "in": "MemToReg", "start":
            {"from": "MEM/WB", "out": "RegBankDst", "to": "ForkDst2", "in": "In", "start": {"
            {"from": "ForkDst2", "out": "Out1", "to": "RegBank", "in": "WriteReg", "points":
            {"from": "ForkDst2", "out": "Out2", "to": "ForwardingUnit", "in": "MEM/WB.Rd"},
            {"from": "MuxMem", "out": "Out", "to": "ForkMemR1", "in": "In", "points": [{"x":
            {"from": "ForkMemR1", "out": "Out1", "to": "RegBank", "in": "WriteData", "points"
            {"from": "ForkMemR1", "out": "Out2", "to": "ForkMEMR2", "in": "In"},
            {"from": "ForkMEMR2", "out": "Out1", "to": "MuxFwdA", "in": "1", "points": [{"x":
            {"from": "MEM/WB", "out": "RegWrite", "to": "ForkRegWR2", "in": "In", "start": {"
            {"from": "ForkRegWR2", "out": "Out1", "to": "RegBank", "in": "RegWrite", "points"
            {"from": "ForkRegWR2", "out": "Out2", "to": "ForwardingUnit", "in": "MEM/WB.RegWr

            {"from": "Control", "out":"JumpReg", "to":"ID/EX", "in":"JumpReg", "start": {"x":

            {"from": "ID/EX", "out":"JumpReg", "to":"ForkSelect", "in":"In", "end": {"x": 435
            {"from": "ForkSelect", "out": "Out1", "to": "OrJumpReg", "in": "JumpRegID/EX", "p
            {"from": "ForkSelect", "out": "Out2", "to": "MuxFwdA/PC", "in": "Jral"},

            {"from": "ForkBr1", "out": "Out1", "to": "OrJumpReg", "in": "AndBranch", "end":{"

            {"from": "OrJumpReg", "out": "Out", "to": "ForkBr2", "in":"In", "start":{"x": 455
            {"from": "MuxFwdA", "out": "Out", "to": "ForkJR", "in": "In", "start": {"x": 440,


            {"from": "ForkJR", "out": "Out1", "to": "MuxFwdA/PC", "in": "0", "points": [{"x":

            {"from": "ForkJR", "out": "Out2", "to": "MuxPC2", "in":"0", "points":[{"x": 455,
            {"from": "ForkBr1", "out": "Out3", "to": "MuxPC2", "in":"Branch", "points": [{"x"
            {"from": "EX/MEM", "out": "Target", "to": "MuxPC2", "in": "1", "start": {"x": 565
            {"from": "MuxPC2", "out":"Out", "to":"MuxPC", "in":"1", "points": [{"x":50, "y":



        ],
        "reg_names": ["zero", "at", "v0", "v1", "a0", "a1", "a2", "a3", "t0", "t1", "t2", "t3", "
        "instructions": "jral-pipeline-set.set"
}
```

7.3. Pruebas

7.3.1. Pruebas Uniciclo

```
########################################

# Test 01

addi $t1,$zero,salto
addi $t0,$zero,1
jalr $ra,$t1
vuelta:
addi $t0,$t0,1
addi $t0,$t0,1
addi $t0,$t0,1
addi $t0,$t0,1
# Terminar ejecucion

salto:
addi $t0,$t0,1
addi $t0,$t0,1
addi $t0,$t0,1
jr $ra

########################################

# Test 02

addi $t1,$zero,salto
addi $t0,$zero,1
sw $t0,0($zero)
lw $t0,0($zero)
jalr $ra,$t1
vuelta:
addi $t0,$t0,1
addi $t0,$t0,1
addi $t0,$t0,1
addi $t0,$t0,1
# Terminar ejecucion

salto:
addi $t0,$t0,1
addi $t0,$t0,1
addi $t0,$t0,1
sw $t0,0($zero)
lw $t0,0($zero)
jr $ra


########################################

# Test 03

addi $t1,$zero,salto
addi $t0,$zero,1
beq $zero,$zero,1 # saltea al jalr
jalr $ra,$t1
salto:
addi $t0,$t0,1
addi $t0,$t0,1
beq $zero,$zero,1 # saltea el jump
jr $ra
addi $t0,$t0,1
addi $t0,$t0,1
# Terminar ejecucion

########################################


# Test 04

addi $t1,$zero,salto
addi $t0,$zero,1
beq $zero,$t1,1 # hace el salto
jalr $ra,$t1
# Vuelve con el jr, terminar ejecucion
nop
nop
nop
salto:
addi $t0,$t0,1
addi $t0,$t0,1
beq $zero,$t1,1
jr $ra                 # toma el salto
addi $t0,$t0,1
addi $t0,$t0,1


########################################
```

7.3.2. Pruebas j en pipeline

```
########################################

#Test 01

test:
addi $t0, $zero, 100
beq $t0,$t0, test
j error
error: addi $t1,$zero,1
# No agarro el salto, agarro el branch
# Si $t1 = 1, entonces hubo un error

########################################

#Test 02
```

```
addi $t0, $zero, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
j test
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
# Si $t1 != 0, entonces hubo un error

########################################

#Test 03
addi $t0, $zero, 1
sw $t0,0($zero)
lw $t0,0($zero)
j test
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
# Si $t1 != 0, entonces hubo un error
# La posicion de memoria 0 va a tener un 1

########################################
```

7.3.3. Pruebas jr en pipeline

```
########################################

#Test 01

test:
```

```
addi $t0, $zero, 100
addi $t2, $zero, error
beq $t0,$t0, test
jr $t2
error: addi $t1,$zero,1
# No agarro el salto, agarro el branch
# Si $t1 = 1, entonces hubo un error

########################################

#Test 02

addi $t0, $zero, 1
addi $t2, $zero, test
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
jr $t2
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
# Si $t1 != 0, entonces hubo un error
########################################

#Test 03
addi $t0, $zero, test
sw $t0,0($zero)
lw $t2,0($zero)
jr $t2
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
# Si $t1 != 0, entonces hubo un error
# Cargo el valor de $t2 un paso antes del jr

########################################
```

7.3.4. Pruebas jalr en pipeline

```
########################################

#Test 01

test:
addi $t2, $zero, error
addi $t0, $zero, 100
beq $t0,$t0, test
jalr $t3,$t2
error: addi $t1,$zero,1
# No agarro el salto, agarro el branch
# Si $t1 = 1, entonces hubo un error

########################################

#Test 02

addi $t0, $zero, 1
addi $t2, $zero, test
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
jalr $t3,$t2
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
# Si $t1 != 0, entonces hubo un error

########################################

#Test 03
addi $t0, $zero, test
sw $t0,0($zero)
lw $t2,0($zero)
jalr $t3,$t2
```

```
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
addi $t1,$zero,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
# Si $t1 != 0, entonces hubo un error
# Cargo el valor de $t2 un paso antes del jalr

########################################

#Test 04
addi $t0, $zero, test
sw $t0,0($zero)
lw $t2,0($zero)
jalr $t3,$t2
addi $t1,$zero,1
jalr $ra,$ra
addi $t1,$t1,1
addi $t1,$t1,1
addi $t1,$t1,1
addi $t1,$t1,1
addi $t1,$t1,1
addi $t1,$t1,1
addi $t1,$t1,1
addi $t1,$t1,1
test:
addi $t0, $t0, 1
addi $t0, $t0, 2
addi $t0, $t0, 3
addi $t0, $t0, 4
jalr $ra,$t3
# Si $t1 != 1, entonces hubo un error
# Cargo el valor de $t2 un paso antes del jalr
# Vuelvo a la posición que me guarde en el jalr ($t3)

########################################
```

7.4. Enunciado

66:20 Organización de computadoras
Trabajo práctico 3: Data Path.

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, <u>con fundamentos reales</u>, 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://bitbucket.org/brunonova/drmips/wiki/Home`, o se puede descargar para Ubuntu, ya sea desde el repositorio de Ubuntu (aunque la versión está desactualizada) o autorizando un repositorio externo (ver [2]).

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

1. Implementar la instrucción `j` en el DP `pipeline.cpu`. Verificar que no se produzcan hazards.
2. Implementar la instrucción `jr` (Jump Register) en el DP `unicycle.cpu`.
3. Implementar la instrucción `jr` en el DP `pipeline.cpu`.
4. Implementar la instrucción `jalr` (Jump and Link Register) en el DP `unicycle.cpu`.
5. Implementar la instrucción `jalr` en el DP `pipeline.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 PC tome el valor deseado, y además que en el caso del DP pipeline 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.
- Los DP, los programas de prueba y los conjuntos de instrucciones usados en cada caso.
- Para los datapath de pipeline, explicar cómo se verificó que no hubiera hazards.
- Este enunciado.

9. Fechas de entrega.

La fecha de entrega de este trabajo práctico es el Jueves 4 de Marzo de 2021.

Referencias

[1] DrMIPS, `https://bitbucket.org/brunonova/drmips/wiki/Home`.

[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.