Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
Seminario de Arquitectura de computadoras
Proyecto Final

Profesor: López Arce Delgado Jorge Ernesto

Alumnos:

León García Daniel Ernesto

Torné Garza Ander

Pacheco Romero Victor Manuel

Cholico

Introducción

MIPS (Microprocessor without Interlocked Pipeline Stages) es una arquitectura diseñada para optimizar la segmentación en unidades de control y para facilitar la generación automática de código máquina por parte de los compiladores. MIPS es RISC (computador con repertorio de instrucciones reducido).
Cuenta con una segmentación de instrucciones: técnica de implementación de unidades de control que permite tratar las instrucciones en serie dividiéndolas en fases.
Con un único cauce de ejecución de instrucciones es posible mantener ejecutándose simultáneamente varias instrucciones, cada una en una fase distinta. Existen múltiples fabricantes de microprocesadores con arquitectura MIPS.

Las características del procesador MIPS32 son las siguientes:

- Máquina RISC (computador con repertorio de instrucciones reducido).

- Ancho de palabra y tamaño de los buses: 32 bits.

- Tamaño de los datos en las instrucciones:
  * Byte (b): 8 bits
  * Halfword (h): 16 bits
  * Word (w): 32 bits
  *Doubleword (d): 64 bits

- Arquitectura de carga / almacenamiento:
  *Antes de ser utilizado en una instrucción aritmética, todo dato debe ser cargado previamente en un registro de propósito general.
  *Instrucciones aritméticas con 3 operandos de 32 bits en registros.
- Esquema de bus único para memoria y E/S.

- Modos de funcionamiento: usuario, núcleo (kernel), supervisor y depuración.

Es importante saber que en MIPS, el compromiso elegido por los diseñadores de la arquitectura es guardar todas las instrucciones con la

misma longitud, por lo que el número de bits en una instrucción de MIPS es siempre de 32, lo mismo que la longitud de una palabra.

A los campos MIPS se les da una serie de nombres para su rápida identificación:

- op: operación básica, llamada “opcode”.
- rs: El registro origen para el primer operando.
- rt: El registro origen para el segundo operando.
- rd:El registro de destino, obtiene el resultado.
- shamt: Usado para instrucciones de desplazamiento.
- funct:Función, este campo (código de función) es usado para seleccionar la variante específica del “opcode”.

En MIPS se distingue tres clases de instrucciones:

- TIPO R
- TIPO I
- TIPO J

Tipo R:
Aritméticas-lógicas

Tipo R

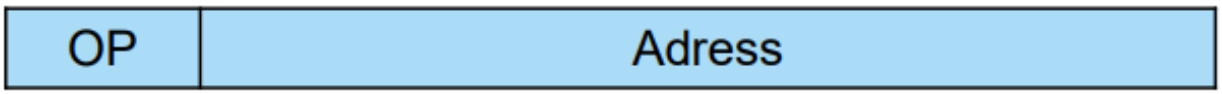
31 : 26	25 : 21	20 : 16	15 11	10 : 6	5 : 0
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Tipo I:
Referencias a memoria, Aritméticas (Inmediato) y Salto condicional.

Tipo I

31 26	25 21	20 16	15 0
op	rs	rt	adress/inmediate
6 bits	5 bits	5 bits	16 bits

Tipo J:
Instrucciones de salto.



Tipo J

31 26	25 0
op	index
6 bits	Adress

Opcode	RS	RT	RD	Shamt	Funct
000000	00000	00001	10000	00000	100000
000000	00100	00010	10001	00000	100001
000000	00011	00011	10010	00000	100010
000000	01110	00111	10011	00000	100011
000000	00010	00011	10100	00000	100100
000000	00011	00001	10101	00000	100101
000000	00101	01010	10111	00000	101010
000000	00000	00000	00000	00000	000000

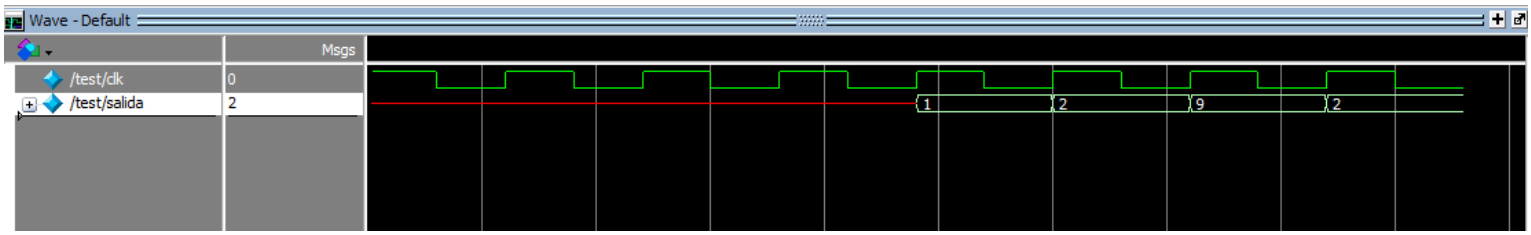
En esta primera fase las operaciones que podrá realizar nuestro diseño son las operaciones aritméticas-lógicas. Un ejemplo de estas son las operaciones básicas, operaciones como OR, AND o otras instrucciones como slt o la nop.

Funct	Operación
100000	Suma
100001	Resta
100010	Multiplicación
100011	División
100100	And
100101	Or
101010	Slt
000000	NoO

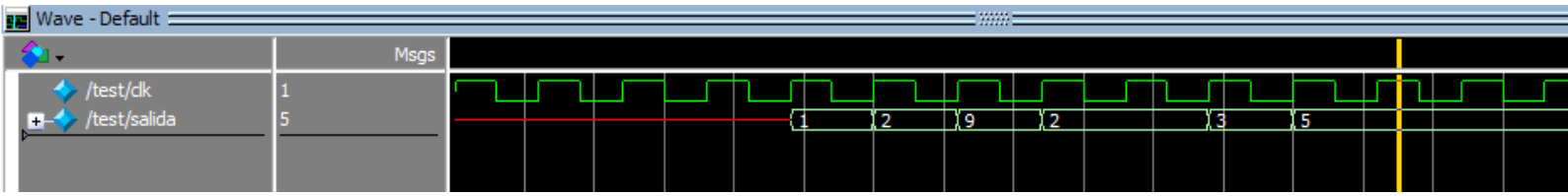
En esta tabla podemos ver la conversión del function a la operación que se va a realizar.

Primeramente, lo que hicimos fue probar si funcionaba con las operaciones básicas, lo que podemos ver en la primera instrucción, es que el function está indicando que se va a realizar una suma ya que el 100000 equivale a la suma. Los valores con los que esta se va a realizar son los que están en la dirección RS y RT, estos siendo 0 y 1 respectivamente, y la dirección en donde estos se van a guardar es donde el RD le indique, este siendo en la dirección 16 del banco de registros, el resultado de esta es uno y se puede comprobar en la imagen a continuación, en la segunda instrucción se realizará una resta de los números 4 y 2, el resultado de esta se guardará en la dirección 17 y tendrá como resultado 2. En la tercera instrucción podemos ver que se realizará una multiplicación de los números 3 y 3, el resultado de esta

operación se guardará en la dirección 18 y tendrá como resultado el valor de 9. Por último, se realizará una división entre los números 14 y 7 que se guardará en la dirección 19 y esta tendrá como resultado el número 2. Todas estas operaciones las podemos verificar en la siguiente imagen.



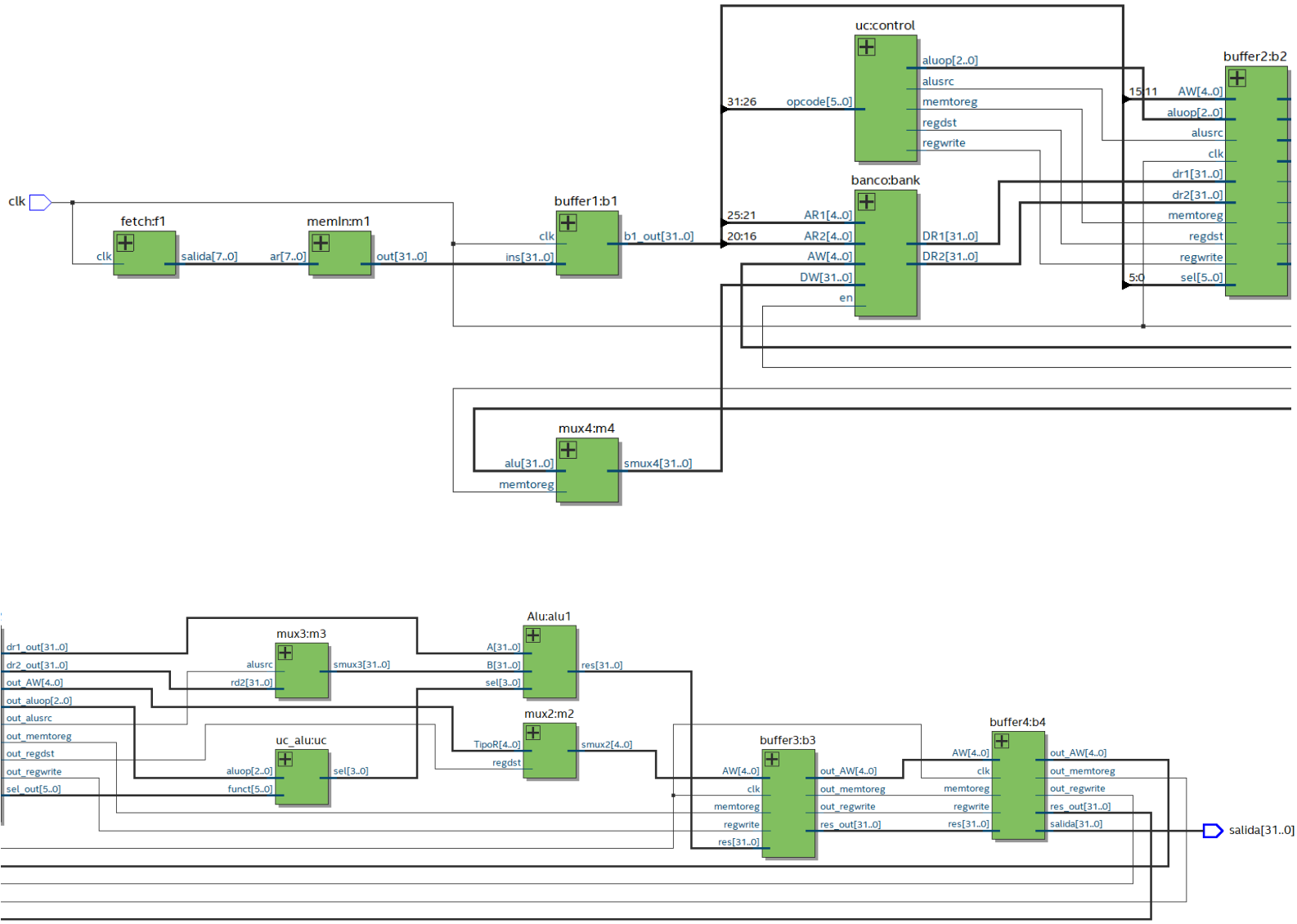
Estos son los datos que se obtienen con las otras cuatro operaciones usando las operaciones AND, OR, Slt y Nop



Y finalmente para comprobar que nuestra fase 1 estuviera correcta, utilizamos los archivos que subió el profesor para comprobar que nos dieran los mismos resultados.

Memory Data - /test/duv/bank/m - Default

0	1	1	2	3	4	5	6	7
8	8	9	10	11	12	13	14	15
16	x	x	x	x	3	1	x	x
24	0	10	1	x	x	x	x	x



La fase uno del proyecto consta de conjunto de módulos que, al conectarlos, estos tienen la capacidad de realizar instrucciones de tipo R, para que este funcione, es necesario implementar los siguientes módulos: el ciclo fetch, la memoria de instrucciones, un banco de

registros, una unidad de control, una ALU control y una ALU. Además de ciertos multiplexores y buffers para que estos funcionen correctamente en conjunto.

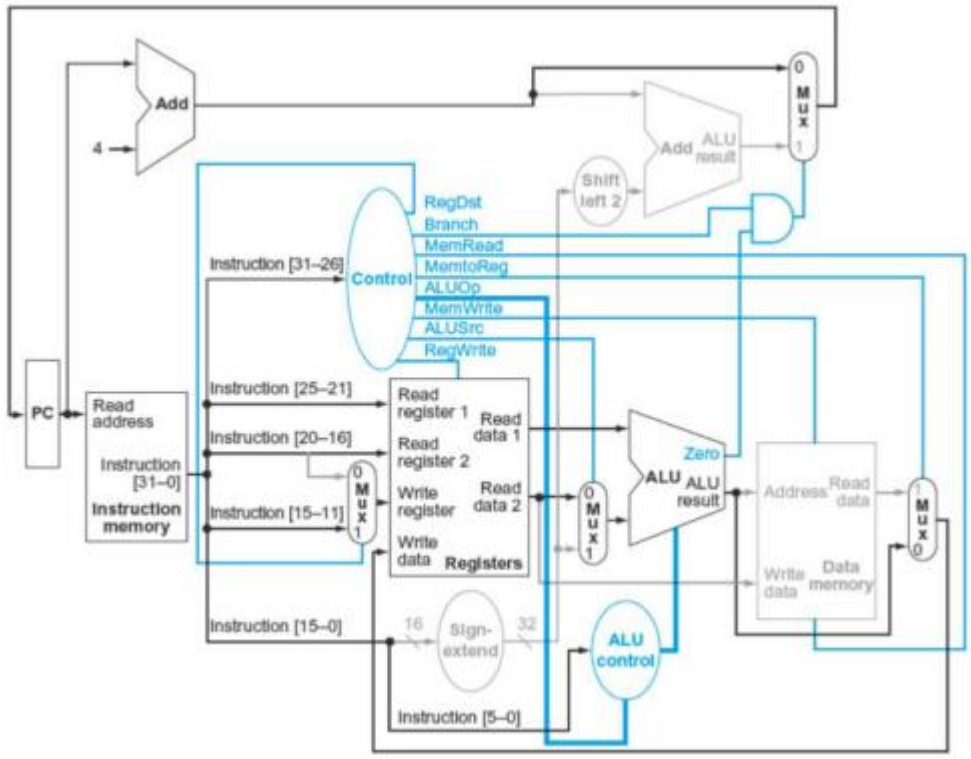
Primeramente, el módulo fetch consta de un contador de programa, un sumador y la memoria de instrucciones. El PC es una memoria de tamaño de palabra de 8 bits, este está inicializado en -4 para que en el primer ciclo este esté en 0 en el que cada que pase un ciclo el sumador le va a ir sumando 4, la memoria de instrucciones, como su nombre lo dice es una memoria en donde se van a guardar las instrucciones con las que se va a trabajar. Esta es una memoria de ancho de palabra de 8 bits y un largo de 129 bits. En esta memoria es donde se guardarán las instrucciones que vamos a ingresar, esto con un $readmemb en donde leerá un archivo de texto en donde están las instrucciones, como las instrucciones son de 32 bits la memoria guarda cada una de estas cada cuatro lugares por lo que en el ciclo fetch cada que pasa un ciclo se le suma cuatro.

Después de esto la instrucción pasa por el primer buffer y de aquí esta se divide, los bits 31:26 se van a la unidad de control, aquí dependiendo del Opcode que sea, esta manda una señal a los módulos de la fase, del bit 25:21 estos siendo el RS se dirigen hacia el Read register 1 del banco de memoria donde más adelante se usarán, del bit 20:16 se van hacia el Read register 2 siendo este el RT y hacia un multiplexor, del bit 15:11 siendo este el RD se dirige hacia el multiplexor y dependiendo del tipo de instrucción que sea pasa un dato o el otro, en este caso como es una instrucción de tipo R, en este caso el RegDst de la unidad de control mandara un 1 por lo que los bits de RD pasarán al Write register. Por último los bits 5:0 se dirigirán hacia la ALU control, aquí la unidad de control mandara la señal de 3 bits 010 por lo que dependiendo del function de la instrucción, la ALU control convertirá la instrucción de 6 bits a una de 4 bits para que este pase a la ALU y le indique qué operación tiene que hacer.

A continuación, podemos ver una tabla con las diferentes operaciones que se pueden realizar en la ALU Control.

Function	ALU	Operación
100000	0000	Suma +
100001	0001	Resta -
100010	0010	Multiplicación *
100011	0011	División /
100100	0100	And &
100101	0101	Or |
100110	0110	Xor ^
100111	0111	Negación ~
101000	1000	Si A<B res = 1 A>B res = 0

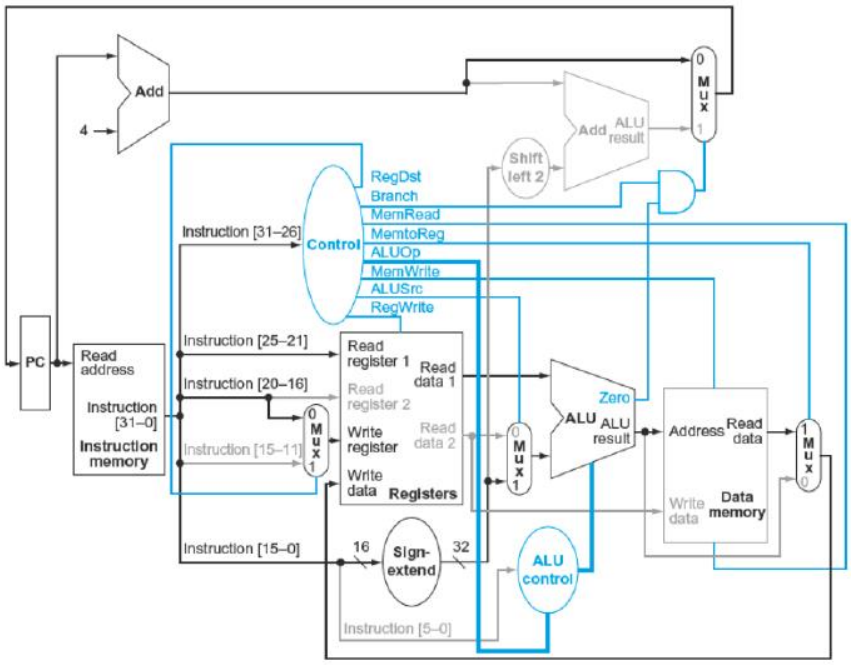
Después de esto obtendremos dos datos como “resultado” en el banco de registros, el read data 1 serán 32 bits que irán directamente a una entrada de la ALU, el dato de read data 2 ira hacia un multiplexor, que cuando este sea una instrucción de tipo R, el ALUSrc sea 0 pasara el valor del RD2 hacia la otra entrada de la ALU, después de esto dependiendo del dato que se obtenga del ALU control, este indicará a la ALU qué operación utilizar, como se observa en la tabla de arriba. Por último el resultado de la operación realizado en la ALU pasa a otro multiplexor, que este al estar en 0, manda el dato hacia el banco de registros en donde se guardará dependiendo de la dirección que contuviera la dirección.



Para la fase 2 del proyecto consta de un conjunto de módulos que hacen que el datapath pueda realizar instrucciones de tipo I, como pueden ser la LW y la BEQ.

Para la realización de las instrucciones LW consta de los siguientes módulos: Ciclo fetch, Unidad de control, Banco de registros, ALU, Alu control y la memoria de datos. Lo primero es que se inicia con el pc que está con un -4 y cada vez que se repita el ciclo se le va a ir sumando 4, pasa por el sumador y por la memoria de instrucciones, al llegar al sumador lo que hace después de sumarle 4 se va al multiplexor que tiene la señal en 0 y regresa nuevamente al pc.Cuando está en la memoria de instrucciones sale y se dividen los bits en 4 partes diferentes, al pasar por el buffer los bits del 31-26 se van a la unidad de control, los bits de 25-21 se van al read register 1 del banco de registros, los bits del 20-16 se van al multiplexor que está con la señal en 0 y los pasa al write register del banco de registros, y los bits del 15-0 se van al sign-extend.

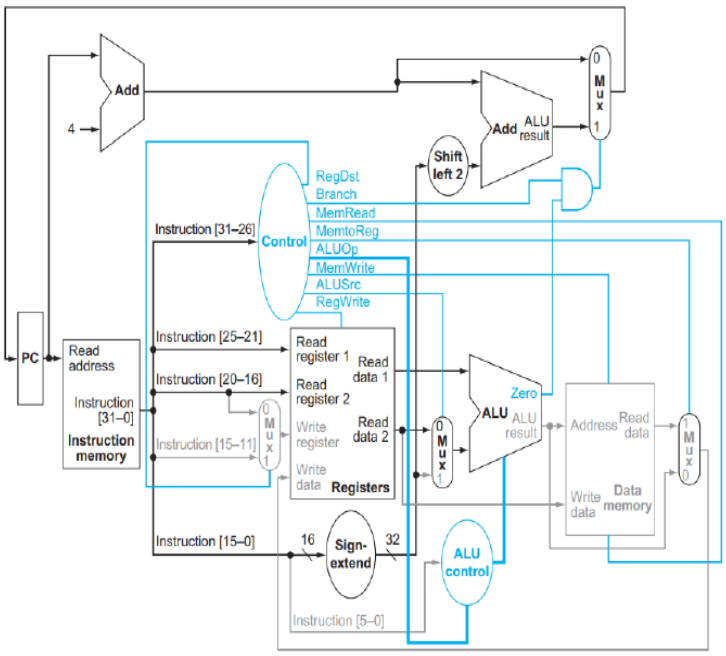
Después obtendremos el “resultado” del banco de registros que es de 32 bits por lo que va directamente a la entrada de la ALU, por el otro lado los 15 bits que están en el sign-extend se convierten en la salida en 32 bits por lo que va directamente a un multiplexor que tiene la señal en 1 por la unidad de control y esto procede a la entrada de la ALU, el alu control mediante la señal que le manda la unidad de control puede saber cuál operación va a ser la que hará y lo que hace es que manda la señal a la ALU, el sign extend lo que hace es que cuando llegan los 16 bits del inmediato le agrega otros 16 bits pero de puros 0, lo que hace que siga siendo el mismo número pero con 32 bits. Después el “resultado” de la alu se va a la memoria de datos y de ahí sale por el read data donde pasa directamente al multiplexor, al estar en el multiplexor si la señal de la unidad de control es 1 manda directamente esos 32 bits hacia el write data del banco de registros.



Para la realización de las instrucciones de tipo I como la BEQ consta de los siguientes módulos: ciclo fetch, unidad de control, banco de registros, sign extend, alu control, sumador, shift left 2.
Iniciamos con el ciclo fetch que ya sabemos como funcione al iniciarlo en -4 y luego le va sumando 4, lo único que cambia es que cuando va del pc al sumado donde se le va agregando 4 va directamente a la entrada del otro sumador y al multiplexor, si el multiplexor está en 0

pues pasa al pc de nuevo pero la unidad de control también manda la señal al 1 que después veremos qué hace, cuando entran los bits a la memoria de instrucciones mediante read address, sale con 32 bits y esos bits pasan al buffer donde se dividen los bits, del 31-26 se van a la unidad de control, 25-21 van hacia la entrada del banco de registros llamada Read register 1 , los bits del 20-16 van hacia el read register 2 y por último los bits del 15-0 van hacia el sign extend.

En el banco de registros salen por el read data 1 con 32 bits directo hacia el otro buffer y de ahí salen junto con la otra salida del banco de registro read data 2 , el read data 1 va directamente a una entrada de la ALU y la otra salida va directamente al multiplexor donde la señal que manda la unidad de control es 0 por lo que pasa directamente a la otra entrada de la ALU, los bits del sign extend ya convertidos en 32 bits va directamente hacia el shift left 2 y esa va directamente a la otra entrada del sumador de al inicio, esto sale por la salida llamada “ALU result” va directamente al multiplexor, debido a esto en la ALU se hace la operación y con el zero flag va directamente a una compuerta lógica AND que va conectada también al BRANCH que viene de la unidad de control, por lo que la salida de esta compuerta va directamente al multiplexor del que manda una señal en 1 y hace que el resultado de “ALU result” vaya directamente hasta el pc de nuevo.



Referencias:

https://previa.uclm.es/profesorado/licesio/Docencia/ETC/21_MIPS-Introduccion-itis.pdf