Tema 3:
Implementación de un procesador no segmentado

Arquitectura de Computadores

Grado en Ingeniería Informática

Mercedes Rodríguez García

Índice

Bibliografía

Estructura y diseño de computadores: Capítulo 4.
Patterson y Hennessy. Editorial Reverte, 2011.

Índice

1. Pasos para implementar un procesador

1. **Analizar** el repertorio de instrucciones que se va a implementar.
2. **Determinar** los componentes necesarios para ejecutar cada tipo de instrucción.
3. **Construir** el camino de datos.
4. **Identificar** las señales de control.
5. **Diseñar** la unidad de control.

Índice

1.1. Análisis del repertorio de instrucciones

Vamos a implementar un procesador **MIPS-32** básico basado en el siguiente repertorio de instrucciones:

1. Instrucciones de **acceso a memoria**

LW	-cargar una palabra-	**LW $t1, offset($t2)**
SW	-almacenar una palabra-	**SW $t1, offset($t2)**

2. Instrucciones **aritmético-lógicas**

ADD	-suma-	**ADD $t1, $t2, $t3**
SUB	-resta-	
AND		
OR		
SLT	-comparación: si menor que-	

3. Instrucciones de **salto**

BEQ	-saltar si es igual-	**BEQ $t1, $t2, offset**
J	-saltar-	**J dirección**

1.1. Análisis del repertorio de instrucciones

En esta máquina, la ejecución de una instrucción se realiza en **5 pasos:**

IF 1º.- Buscar en la memoria de instrucciones (L1 de instrucciones) la instrucción que se va a ejecutar.
2º.- Incrementar el contador del programa (PC) para que apunte a la siguiente instrucción.

ID Decodificar la instrucción y, simultáneamente, leer los registros fuente para obtener los operandos

EX Realizar el cálculo.

MEM Acceder a memoria (cuando sea necesario).

WB Escribir el resultado en un registro del procesador (cuando sea necesario).

Las acciones a realizar en cada paso varían ligeramente de un tipo de instrucción a otro.

1.1. Análisis del repertorio de instrucciones

Acciones a realizar en las **instrucciones de carga**:

LW $t1, offset($t2)

Este es el registro que contiene la dirección base.

Cantidad de posiciones de memoria que hay que desplazarse respecto a la dirección base.

IF	Buscar la instrucción e incrementar el contador del programa (PC).
ID	Leer el registro fuente que contiene la dirección base y decodificar la instrucción.
EX	Calcular la dirección de memoria: **dirección base + offset**
MEM	Acceder a memoria para obtener el dato.
WB	Escribir el dato en el registro destino.

1.1. Análisis del repertorio de instrucciones

Acciones a realizar en las **instrucciones de almacenamiento**:

SW $t1, offset($t2)

Este es el registro que contiene la dirección base.

Cantidad de posiciones de memoria que hay que desplazarse respecto a la dirección base.

IF Buscar la instrucción e incrementar el contador del programa (PC).

ID Leer el registro fuente que contiene la dirección base, leer el registro fuente que contiene el dato que se va a almacenar en memoria y decodificar la instrucción.

EX Calcular la dirección de memoria : **dirección base + offset**

MEM Acceder a memoria para almacenar el dato.

WB ----

1.1. Análisis del repertorio de instrucciones

Acciones a realizar en las **instrucciones de aritmético-lógicas**: **ADD $t1, $t2, $t3**

Este es el registro destino (donde se almacena el resultado).

IF Buscar la instrucción e incrementar el contador del programa (PC).

ID Leer los registros fuentes que contienen los dos operandos y decodificar la instrucción.

EX Realizar la operación aritmético-lógica: **operandoX + operandoY**

MEM ----

WB Escribir el resultado en el registro destino.

1.1. Análisis del repertorio de instrucciones

Acciones a realizar en las **instrucciones de salto condicional**:

BEQ $t1, $t2, offset

Número de instrucciones que hay que saltar en el programa (si se salta hacia delante el número es positivo, si se salta hacia atrás el número es negativo).

IF	Buscar la instrucción e incrementar el contador del programa (PC).
ID	Leer los registros fuentes para obtener los dos operandos que se van a comparar y decodificar la instrucción.
EX	Realizar la comparación mediante una resta: **operandoX - operandoY** y calcular la dirección destino del salto: **PC + offset * 4**
MEM	Si la comparación es cierta, modificar PC con la dirección destino del salto.
WB	----

1.1. Análisis del repertorio de instrucciones

Acciones a realizar en las **instrucciones de salto incondicional**: **J dirección**

IF	Buscar la instrucción e incrementar el contador del programa (PC).
ID	Decodificar la instrucción.
EX	Calcular la dirección destino del salto: **4bits más significativos del PC . campo dirección . 00_2**
MEM	Modificar PC con la dirección destino del salto.
WB	----

1.1. Análisis del repertorio de instrucciones

	LW	SW	ADD	BEQ	J
IF	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC
ID	Leer un registro y decodificar	Leer dos registros y decodificar	Leer dos registros y decodificar	Leer dos registros y decodificar	----- y decodificar
EX	Calcular dirección de memoria (suma)	Calcular dirección de memoria (suma)	Calcular operación	Comparar operandos (resta) y calcular dirección del salto (suma)	Calcular dirección del salto (concatenación)
MEM	Leer dato	Escribir dato	-----	Actualizar PC con la dirección destino del salto	Actualizar PC con la dirección destino del salto
WB	Escribir el dato obtenido de memoria en un registro	-----	Escribir resultado del cálculo en un registro	-----	-----

1.1. Análisis del repertorio de instrucciones

1

LW	**LW $t1, offset($t2)**
SW	**SW $t1, offset($t2)**

código	$R_{Fuente-1}$	$R_{Fuente-2}$ $R_{Destino}$	offset
31-26	**25-21**	**20-16**	**15-0**

2

ADD	**ADD $t1, $t2, $t3**
SUB	
AND	
OR	
SLT	

código	$R_{Fuente-1}$	$R_{Fuente-2}$	$R_{Destino}$	----	$código_{func}$
31-26	**25-21**	**20-16**	**15-11**	**10-6**	**5-0**

3

BEQ	**BEQ $t1, $t2, offset**
J	**J dirección**

código	$R_{Fuente-1}$	$R_{Fuente-2}$	offset
31-26	**25-21**	**20-16**	**15-0**

código	dirección
31-26	**25-0**

1.1. Análisis del repertorio de instrucciones

	LW	SW	ADD	BEQ	J
IF	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC
ID	Leer un registro **(bits 25-21)** y decodificar	Leer dos registros **(bits 25-21 y 20-16)** y decodificar	Leer dos registros **(bits 25-21 y 20-16)** y decodificar	Leer dos registros **(bits 25-21 y 20-16)** y decodificar	----- y decodificar
EX	Calcular dirección de memoria (suma)	Calcular dirección de memoria (suma)	Calcular operación	Comparar operandos (resta) y calcular dirección del salto (suma)	Calcular dirección del salto (concatenación)
MEM	Leer dato	Escribir dato	-----	Actualizar PC con la dirección destino del salto	Actualizar PC con la dirección destino del salto
WB	Escribir el dato obtenido de memoria en un registro **(bits 20-16)**	-----	Escribir resultado del cálculo en un registro **(bits 15-11)**	-----	-----

1.1. Análisis del repertorio de instrucciones

	LW	SW	ADD	BEQ	J
IF	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC	Buscar instrucción e incrementar PC
ID	Leer un registro **(bits 25-21)** y decodificar	Leer dos registros **(bits 25-21 y 20-16)** y decodificar	Leer dos registros **(bits 25-21 y 20-16)** y decodificar	Leer dos registros **(bits 25-21 y 20-16)** y decodificar	----- y decodificar
EX	Calcular dirección de memoria (suma) **dirección base + offset**	Calcular dirección de memoria (suma) **dirección base + offset**	Calcular operación **operandoX + operandoY**	Comparar operandos (resta) **operandoX - operandoY** y calcular dirección del salto (suma) **PC + offset * 4**	Calcular dirección del salto (concatenación) **4bits más significativos del PC . campo dirección . 00_2**
MEM	Leer dato de memoria	Escribir dato en memoria	-----	Actualizar PC con la dirección destino del salto BEQ	Actualizar PC con la dirección destino del salto J
WB	Escribir el dato obtenido de memoria en un registro **(bits 20-16)**	-----	Escribir resultado del cálculo en un registro **(bits 15-11)**	-----	-----

1.1. Análisis del repertorio de instrucciones

Se va a ejecutar la instrucción ADD $1, $2, $3.
1.- Convierte la instrucción a binario.
2.- Describe qué realiza cada fase de ejecución.
3.- ¿Qué valores contendrán los registros $1, $2 y $3 cuando termine de ejecutarse la instrucción?

Banco de registros del procesador

$0	24	44	5	89	1	D2	2	2301
$8	2	9	0	0	4	44	67	32
$16	4	7	67	0	5	60	0	55
$24	9	8	5	7	6	34	4	5C2

1.1. Análisis del repertorio de instrucciones

Se va a ejecutar la instrucción 00010000100001100000000000001000.
1.- Describe qué realiza cada fase de ejecución.

Banco de registros del procesador

$0	24	44	5	89	1	D2	1	2301
$8	24	9	0	0	4	44	67	32
$16	4	7	67	0	5	60	0	55
$24	9	8	5	7	6	34	4	5C2

1.1. Análisis del repertorio de instrucciones

Se va a ejecutar la instrucción 10001100000000010000000000000100.

1.- Describe qué realiza cada fase de ejecución.

2.- Cuando termine de ejecutarse la instrucción, ¿algún registro del banco habrá cambiado de valor?

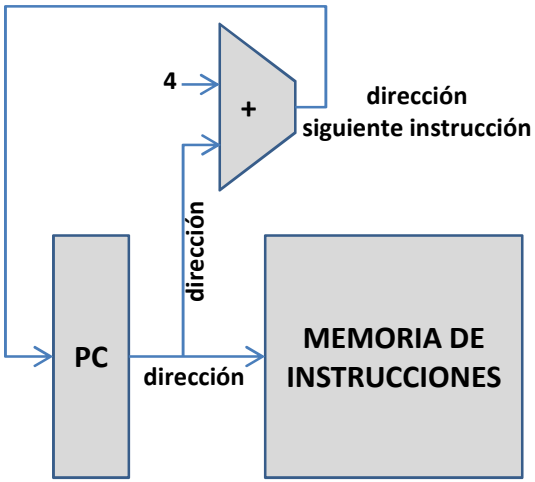
Banco de registros del procesador

$0	24	44	5	89	1	D2	1	2301
$8	2	9	0	0	4	44	67	32
$16	4	7	67	0	5	60	0	55
$24	9	8	5	7	6	34	4	5C2

Índice

1.2. Identificación de las unidades funcionales

IF



Unidades funcionales:

- **Memoria de instrucciones:** lugar donde se almacenan las instrucciones del programa. Se corresponde con la memoria caché L1 de instrucciones. El procesador nunca va a escribir en esta memoria, siempre realizará lecturas.

- **Contador de programa (PC):** registro de 32 bits que contiene la dirección de la instrucción que se va a ejecutar.

- **Sumador:** circuito combinacional encargado de calcular la dirección de la siguiente instrucción.

¿Por qué se incrementa el PC en cuatro unidades?

1.2. Identificación de las unidades funcionales

ID

Unidades funcionales:

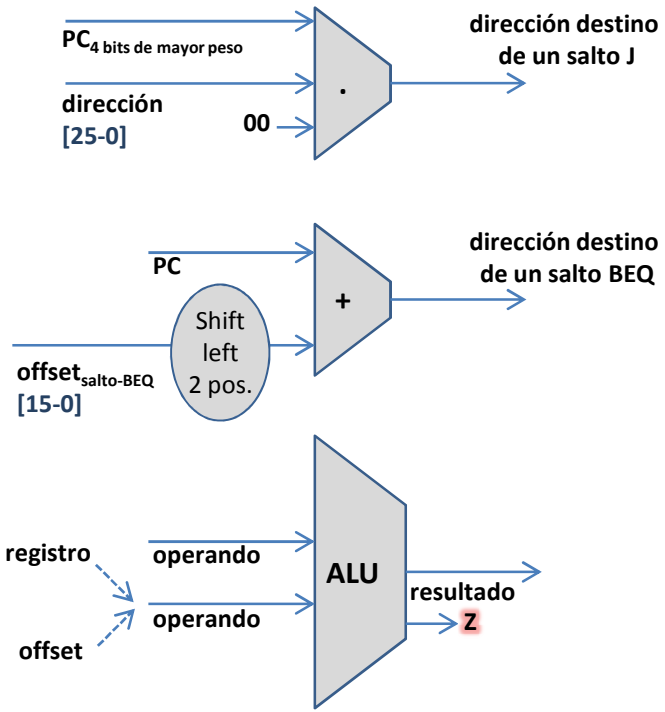
- **Banco de registros:** colección de 32 registros de 32 bits. Para acceder a un registro concreto (ya sea para leer o escribir) hay que especificar su número de identificación. Este número de identificación lo proporciona la instrucción.

NOTA: el elemento que decodifica una instrucción es el decodificador, pero no lo vamos considerar porque pertenece a la unidad de control y no al camino de datos.

¿Cuántos bits tienen las entradas de los identificadores de registro? ¿Por qué?

1.2. Identificación de las unidades funcionales

EX



¿Para qué sirve el componente Shift Left de 2 posiciones?

Unidades funcionales:

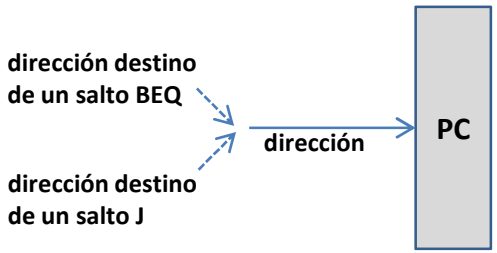
- **Concatenador:** circuito combinacional encargado de realizar la concatenación que obtiene la dirección destino de un salto incondicional (J). Por su simplicidad se implementa como circuito independiente de la ALU.

- **Sumador:** circuito combinacional encargado de realizar la suma que calcula la dirección destino de un salto condicional (BEQ). Se implementa como circuito independiente de la ALU para poder realizar este cálculo simultáneamente con la comparación que tiene lugar en las instrucciones BEQ.

- **ALU:** unidad aritmético-lógica que tiene dos entradas de 32 bits y produce un resultado de 32 bits, así como una señal de un bit (Z) que se activa cuando el resultado es cero. Observamos que, dependiendo del tipo de instrucción, la segunda entrada de la ALU puede provenir de un registro o directamente del campo offset de la instrucción.

1.2. Identificación de las unidades funcionales

MEM



Unidades funcionales:

- **PC:** si la instrucción es un salto hay que sobreescribir el contador de programa con la dirección destino del salto. La dirección destino puede provenir de dos unidades funcionales diferentes (sumador en los saltos BEQ, concatenador en los saltos J) .

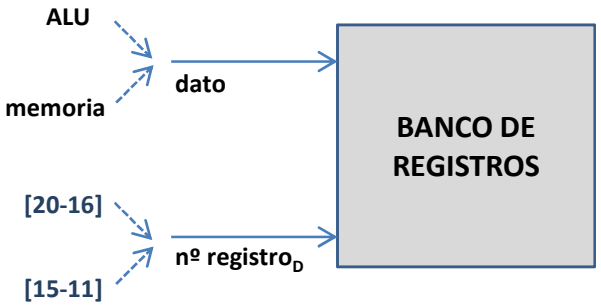
- **Memoria de datos:** lugar donde se almacenan los datos del programa. Se corresponde con la memoria caché L1 de datos.

Tiene dos entradas, por una se suministra la dirección de la posición que se va a leer o escribir y, por otra, si la instrucción es de almacenamiento, se suministra el dato que se va a escribir.

Tiene una salida que proporcionará el dato que se ha leído cuando la instrucción es de carga.

1.2. Identificación de las unidades funcionales

WB



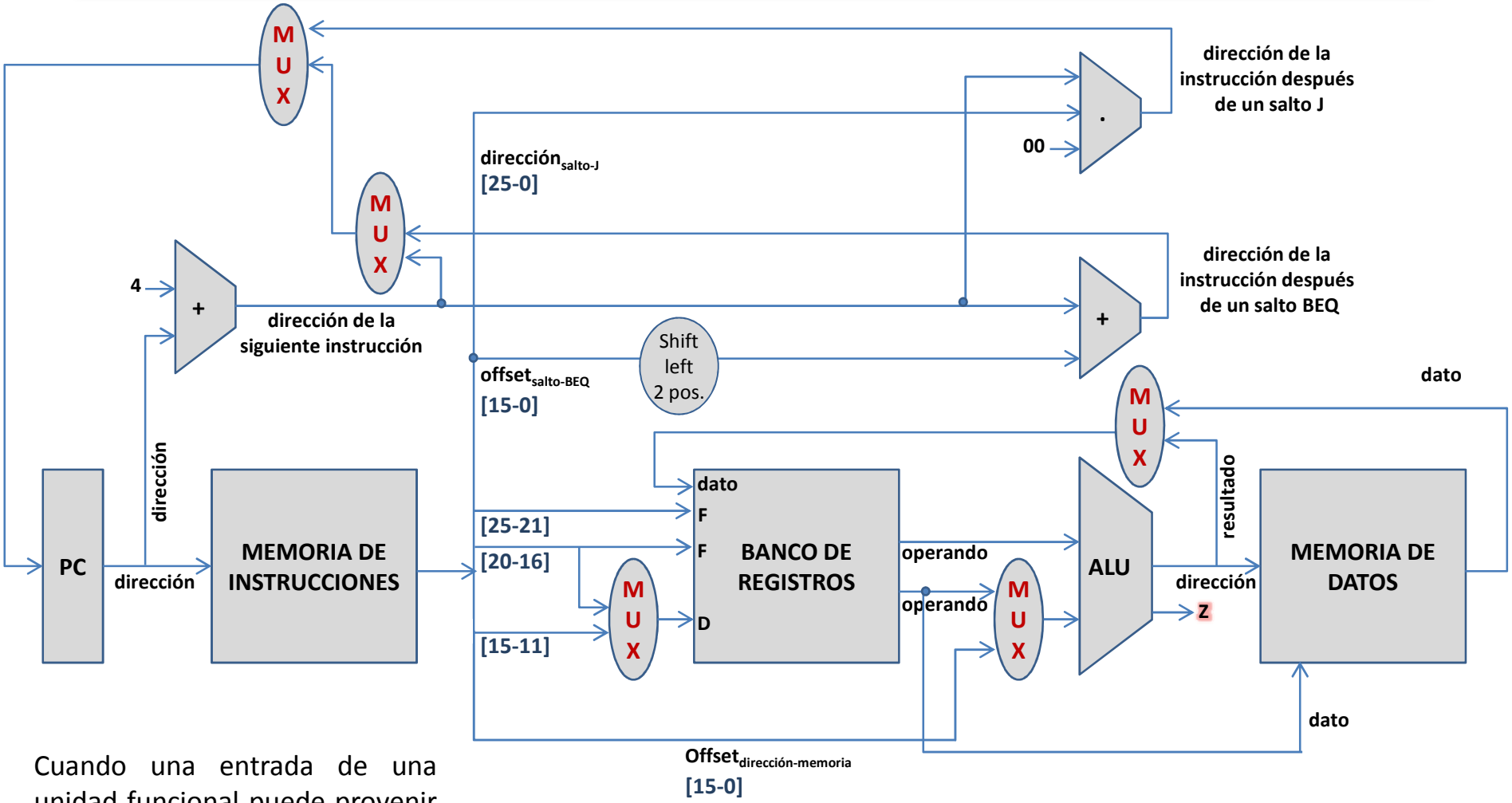
Unidades funcionales:

- **Banco de registros:** El valor que se guardará en el registro destino puede provenir de la ALU (si la instrucción es aritmético lógica) o de la memoria (si la instrucción es de carga).

Por otro lado, el número de identificación del registro destino puede estar en dos campos diferentes de la instrucción. En las instrucciones de carga está en los bits 20-16 y en las aritmético-lógicas está en los bits 15-11.

Índice

1.3. Construcción del camino de datos



Cuando una entrada de una unidad funcional puede provenir de dos orígenes diferentes se utilizan multiplexores.

Índice

1.4. Identificación de las señales de control

La unidad de control (UC) recibe como entrada el código de la instrucción **[31-26]** (y a veces también el código de función), lo decodifica y, en función del tipo de instrucción, genera las señales de control necesarias para ejecutarla.

Estas señales serán las que controlen las operaciones de escritura, los multiplexores y la ALU.

1.4. Identificación de las señales de control

Control de <u>las operaciones de escritura</u> …

1 **¿Por qué hay que temporizar las escrituras en un registro?**

Hay que temporizar las escrituras en un registro porque **si un dato se escribe en el mismo instante en que es leído**, el valor leído puede corresponder al valor antiguo, al valor nuevo o a una combinación de ambos. **-[PATT11]-**

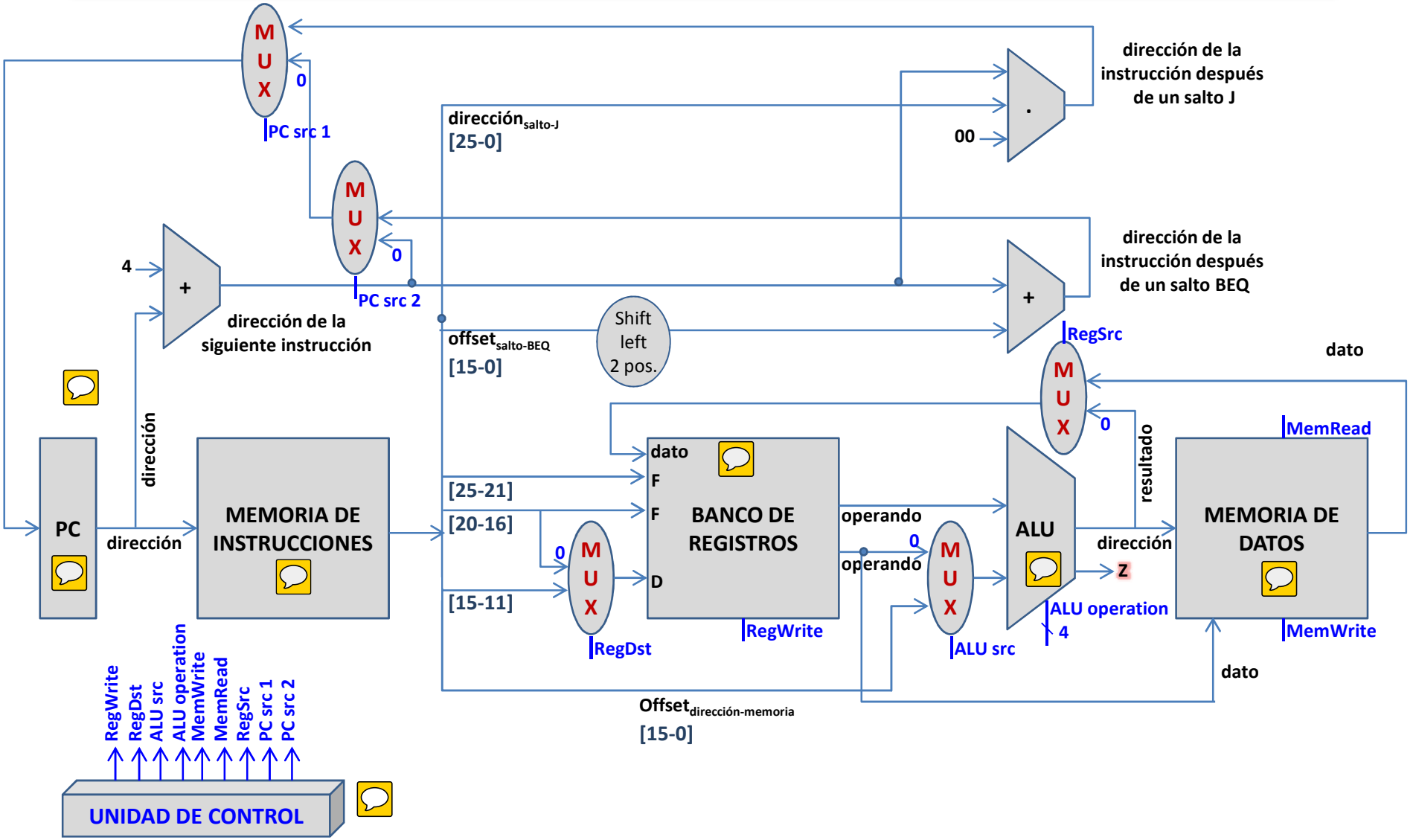
2 **¿Cuándo se puede realizar la escritura en un registro?**

Las escritura en un registro sólo se pueden realizar en los **flancos del reloj**. Es indiferente trabajar con flanco de subida o flanco de bajada, nosotros trabajaremos con flaco de subida.

3 **¿Se requiere alguna señal de control?**

Sí, **para indicar en qué ciclo se realizará la escritura**. Excepción: no se necesitará señal de control cuando la escritura en un registro se tenga que efectuar en TODOS LOS CICLOS.

1.4. Identificación de las señales de control



1.4. Identificación de las señales de control

RegWrite.- Controla cuándo se realiza una escritura en el banco de registros. Debe estar activa para que la escritura se lleve a cabo en el flanco de subida del reloj.

RegDst.- Determina de dónde proviene el número de identificación del registro destino. Si está activada el número de identificación proviene del campo de la instrucción que ocupa los bits 15-11 (instrucciones aritmético-lógicas) y si está desactivada del campo que ocupa los bits 20-16 (instrucciones de carga) .

RegSrc.- Determina de dónde proviene el dato que se va a almacenar en el registro destino. Si está activada será el que provenga de memoria y si está desactivada el que provenga de la ALU.

PC src 1.- Determina si la dirección de la siguiente instrucción proviene de un salto incondicional J.

PC src 2.- Determina si la dirección de la siguiente instrucción proviene de un salto condicional BEQ.

1.4. Identificación de las señales de control

ALU src.- Determina de dónde proviene el segundo operando de la ALU. Si la señal está activada proviene del campo offset de la instrucción (instrucciones de carga y almacenamiento), si está desactivada proviene de un registro (instrucciones aritmético-lógicas y saltos condicionales).

ALU operation.- Señal de 4 bits utilizada para seleccionar la operación que realizará la ALU.

Instrucción	ALU operation	Operación
LW	0010	Suma
SW	0010	Suma
ADD	0010	Suma
SUB	0110	Resta
AND	0000	AND
OR	0001	OR
SLT	0111	Comparación
BEQ	0110	Resta

1.4. Identificación de las señales de control

MemRead.- Controla cuándo se realiza una lectura en la memoria de datos. Si está activada realiza la lectura, si está desactivada no hace nada.

MemWrite.- Controla cuándo se realiza una escritura en la memoria de datos. Si está activada realiza la escritura, si está desactivada no hace nada.

NOTA: MemWrite y MemRead nunca pueden estar activas a la vez.

¿Por qué la memoria de instrucciones no tiene ninguna señal de control?

Índice

1.5. Diseño de la unidad de control

La unidad de control (UC) es la encargada activar y desactivar las señales de control en función del código de instrucción (bits 31-26) y, en determinados casos, también del código de función (bits 5-0) .

Instrucción	Código de instrucción (bits 31-26)
LW	100011
SW	101011
ADD	000000
SUB	000000
AND	000000
OR	000000
SLT	000000
BEQ	000100
J	000010

Instrucción	Código de función (bits 5-0)
ADD	100000
SUB	100010
AND	100100
OR	100101
SLT	101010

1.5. Diseño de la unidad de control

La señal de control ALU operation (4 bits) identifica la operación que hay que realizar en la ALU.

Instrucción	ALU operation	Operación
LW	0010	Suma
SW	0010	Suma
ADD	0010	Suma
SUB	0110	Resta
AND	0000	AND
OR	0001	OR
SLT	0111	Comparación
BEQ	0110	Resta

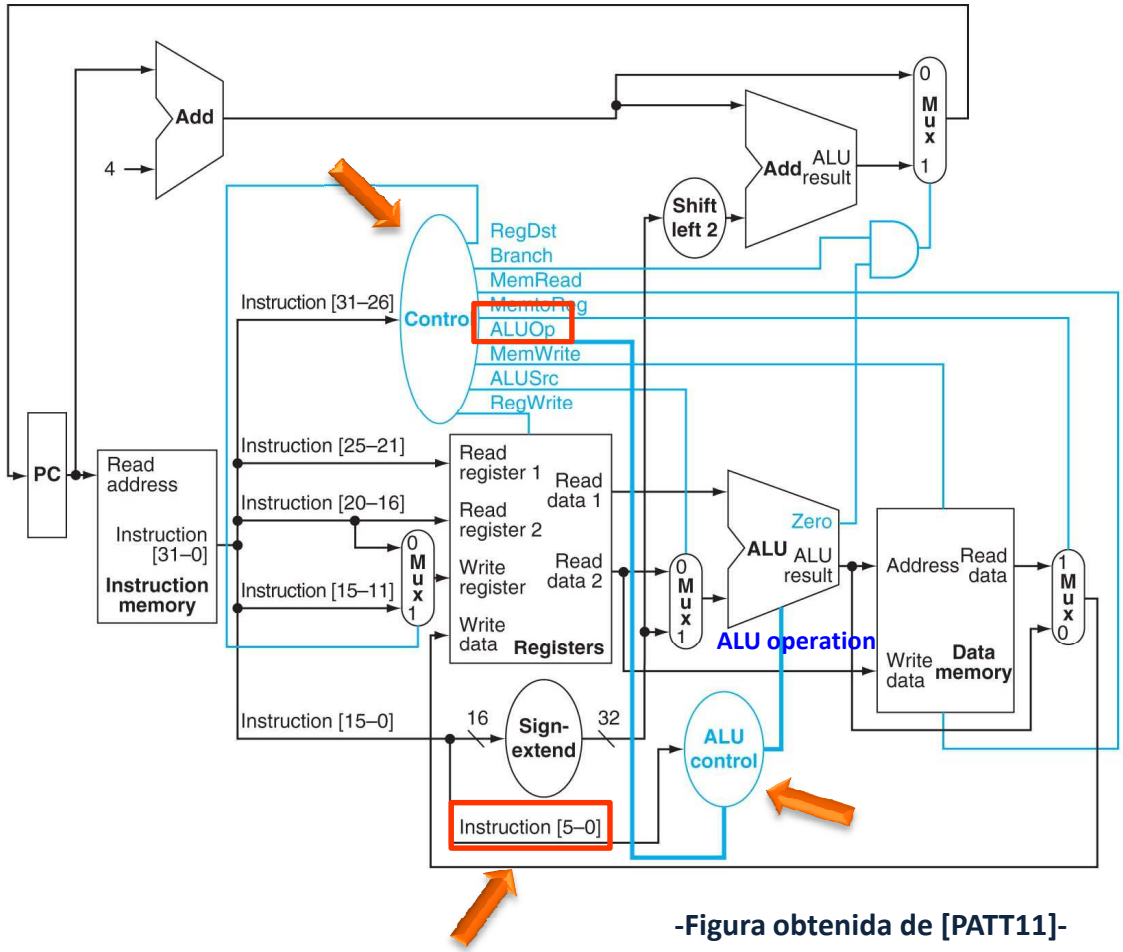
1.5. Diseño de la unidad de control

La lógica de la unidad de control queda definida como una gran tabla de verdad donde las entradas son los 6 bits del código de instrucción (bits 31-26) y, en algunos casos, los 6 bits del código de función (bits 5-0) y las salidas son las líneas de control. **-[PATT11]-**

	ENTRADAS		SALIDAS								
Instrucción	Código operación	Código función	**RegWrite**	**RegDst**	**RegSrc**	**PC src 1**	**PC src 2**	**ALU src**	**ALU operation**	**MemRead**	**MemWrite**
LW	100011	--							0010		
SW	101011	--							0010		
ADD	000000	100000	1	1	0	0	0	0	0010	0	0
SUB	000000	100010							0110		
AND	000000	100100							0000		
OR	000000	100101							0001		
SLT	000000	101010							0111		
BEQ	000100	--							0110		
J	000010	--							---		

Índice

2. Implementación del procesador de laboratorios



-Figura obtenida de [PATT11]-

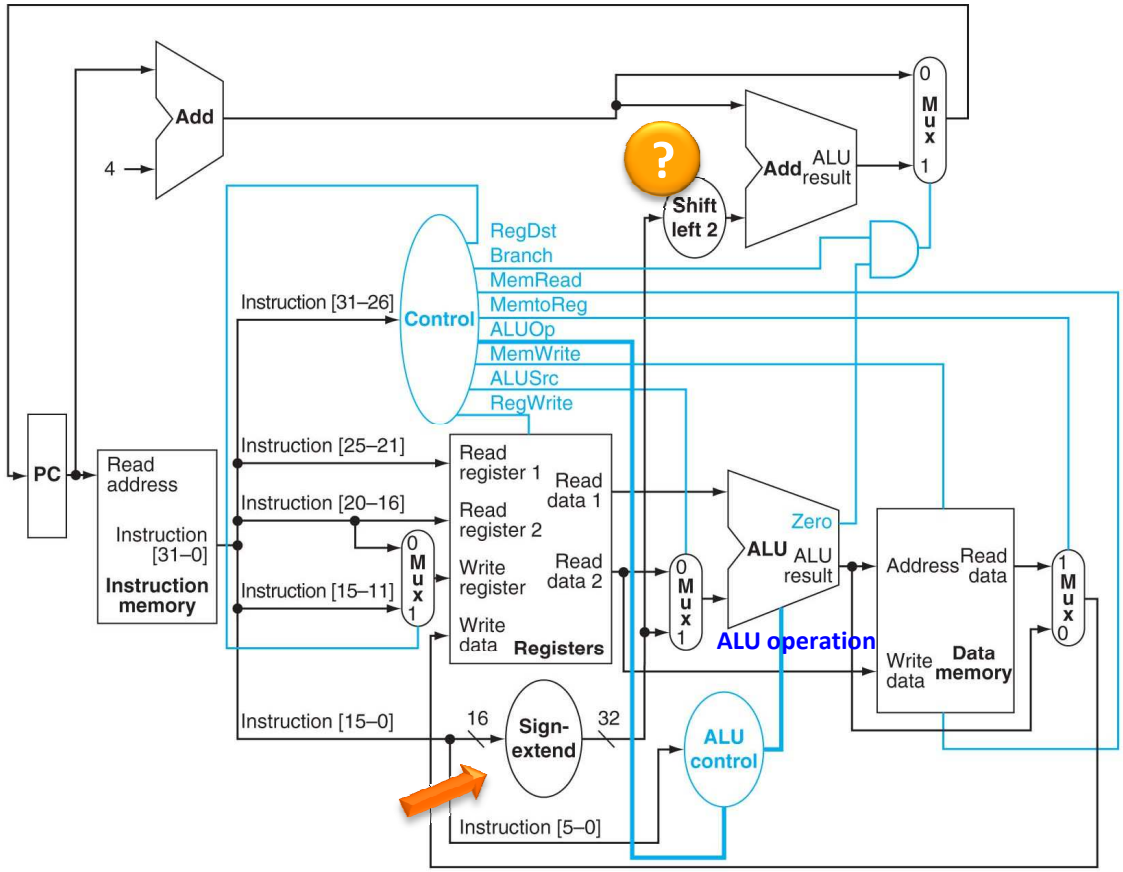
Novedad:

- **Unidad de control distribuida:**

Es muy común encontrarnos procesadores con una unidad de control principal y varias unidades de control más pequeñas distribuidas por el procesador. Es una optimización que incrementa la velocidad de la unidad de control.

Por ejemplo, la pequeña unidad de control de la ALU es la que genera los 4 bits de la señal ALU Operation. Para ello, recibe como entradas el código$_{FUNC}$ de la instrucción (en el dibujo aparece referenciado por Instruction [5-0]) y una señal de control adicional llamada ALUOp generada a su vez por la unidad de control principal.

2. Implementación del procesador de laboratorios

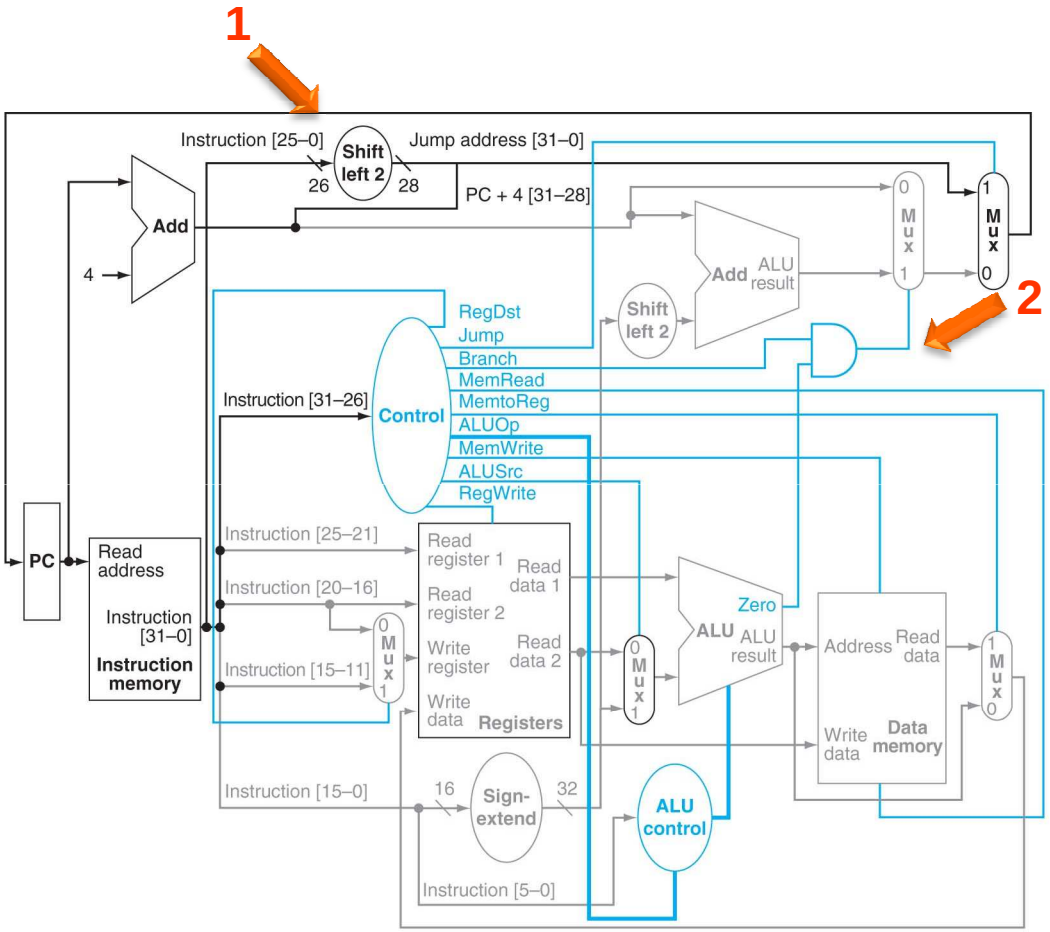


-Figura obtenida de [PATT11]-

Novedad:

- **Unidad de extensión de signo:** la unidad etiquetada como *Sign-extended* es un circuito que incrementa el tamaño de un dato mediante la replicación del bit de signo. En nuestro caso, el dato de entrada es de 16 bits y lo extenderá a 32 bits.

2. Implementación del procesador de laboratorios



-Figura obtenida de [PATT11]-

Novedad:

- **Interior del concatenador:** el punto **1** muestra el interior del concatenador del salto J. La unidad etiquetada como shift left *2* es un dasplazador que añade 00_2 en la parte menos significativa.

- **Puerta AND:** la puerta AND del punto **2** forma parte de la unidad de control distribuida. Genera la señal de control **PC src 2**. Esta señal indica si hay que sobrescribir el PC en un salto BEQ.