

# Implementación de la ISA

Parte 1

OdC 2020

# Ejercicio 1

Marque con una barra invertida e indique el número de bits que representa cada una de las líneas del *data path*.

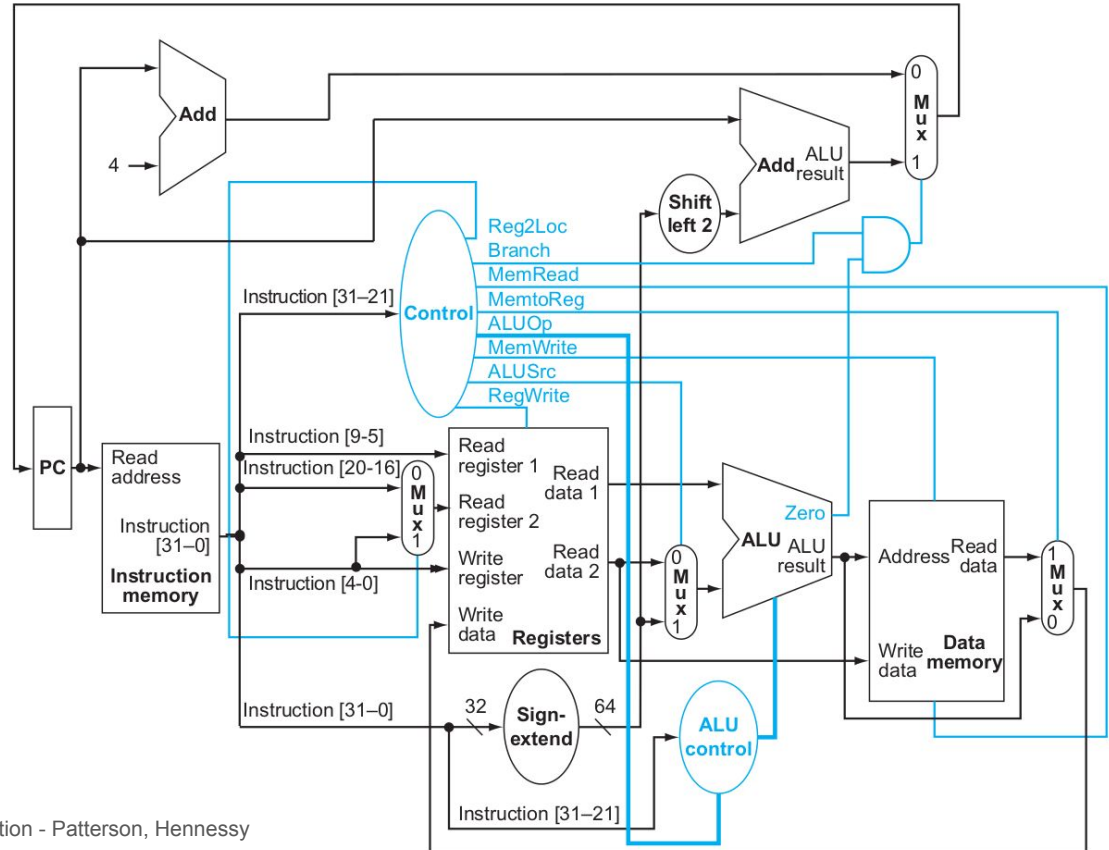
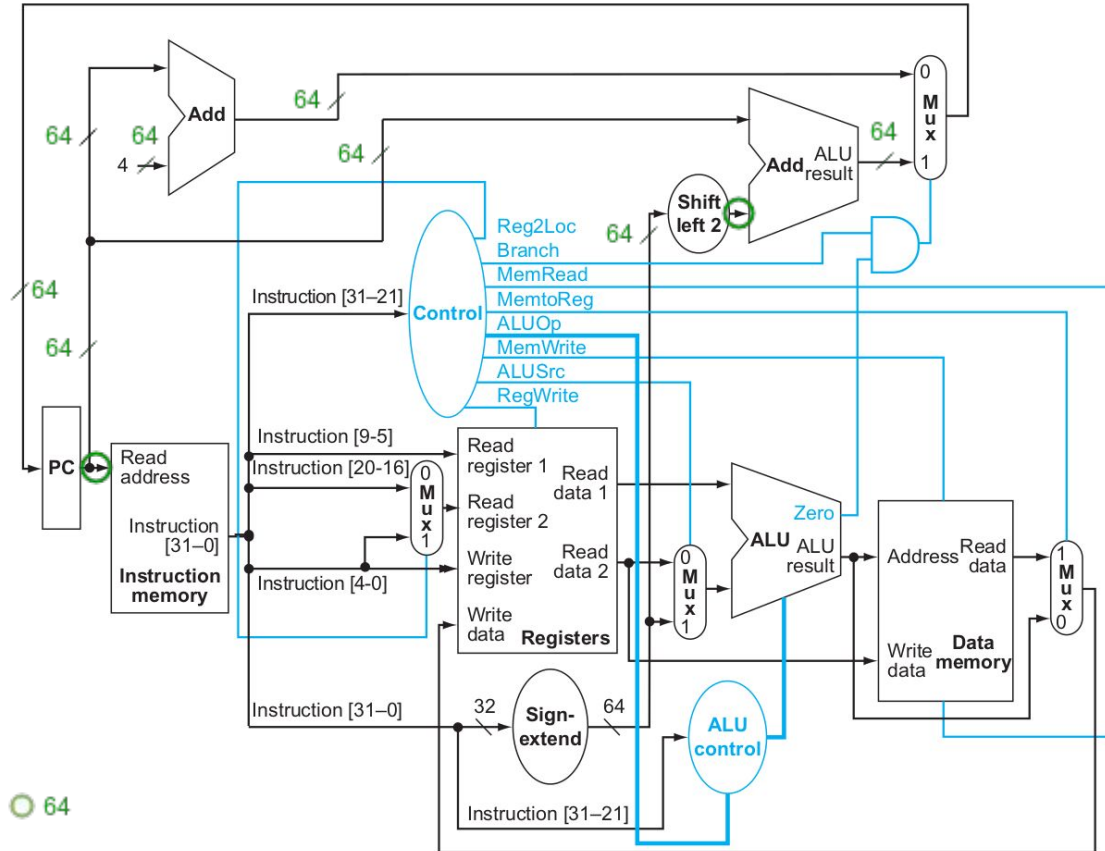
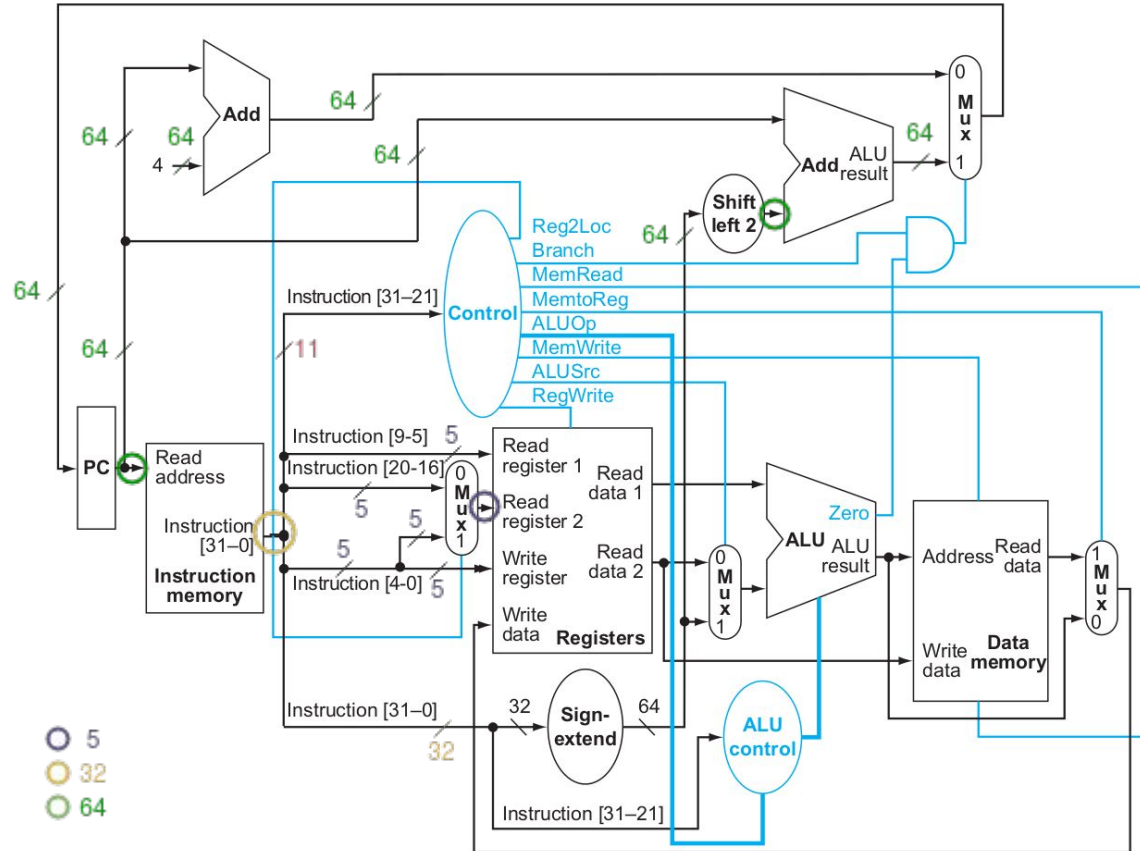


Fig 4.17 - Computer Organization and Design, ARM Edition - Patterson, Hennessy

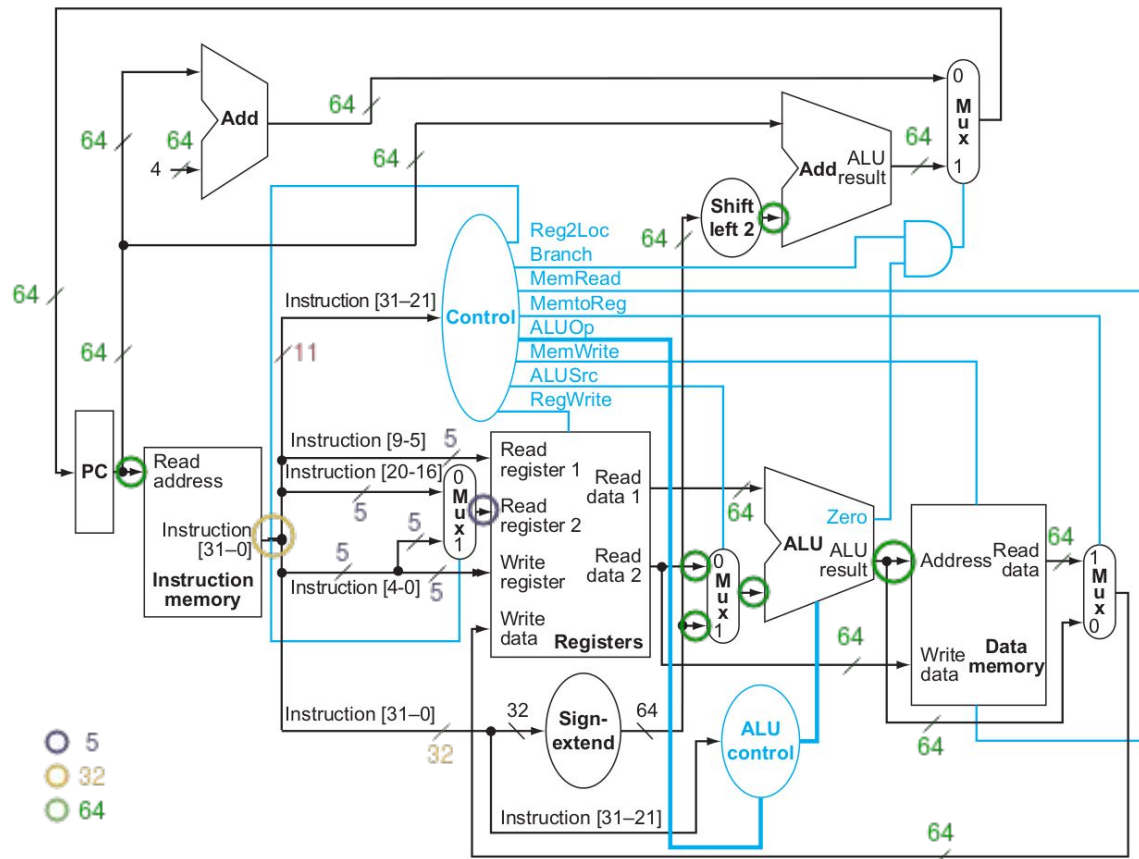
# Ejercicio 1



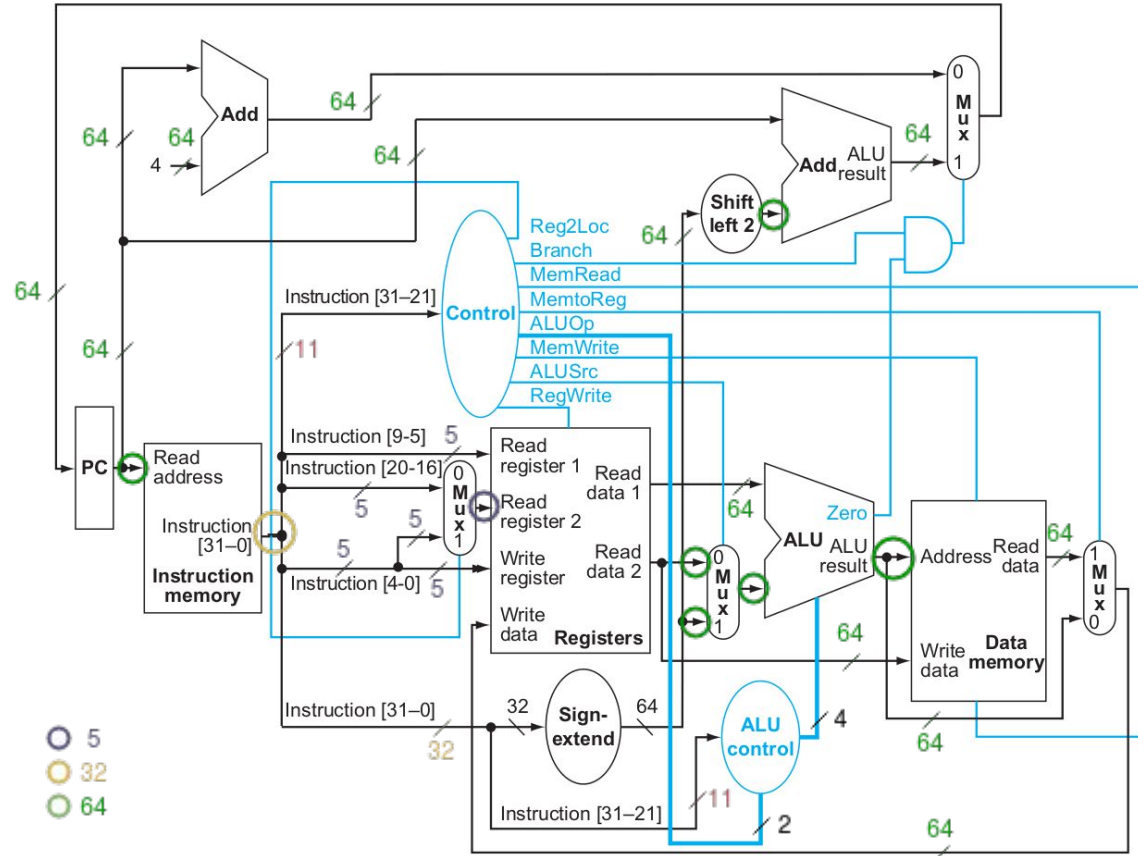
# Ejercicio 1



# Ejercicio 1



# Ejercicio 1



# Ejercicio 2

Considerando la siguiente distribución de instrucciones en un programa:

R-type	I-Type	LDUR	STUR	CBZ	B
24%	28%	25%	10%	11%	2%

2.1) Qué porcentaje de todas las instrucciones utiliza `data memory`?

Las instrucciones que acceden a memoria de datos son:

- LDUR
- STUR

Por lo tanto, el 35% de las instrucciones utilizan el recurso `data memory`.

2.2) Qué porcentaje de todas las instrucciones utiliza `instruction memory`?

Todas las instrucciones deben acceder a la memoria de instrucción! (Rta.: 100%)

2.3) Qué porcentaje de todas las instrucciones utiliza el `sign extend`?

Los tipos de instrucción que utilizan el bloque de extensión de signo son:

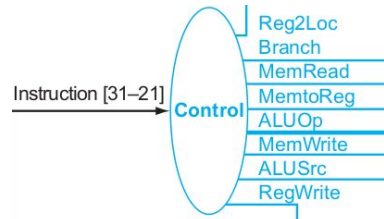
- Tipo I
- Tipo CB
- Tipo B
- Tipo D

Por lo tanto, el 76% de las instrucciones utilizan el recurso `sign extend`.

# Ejercicio 3

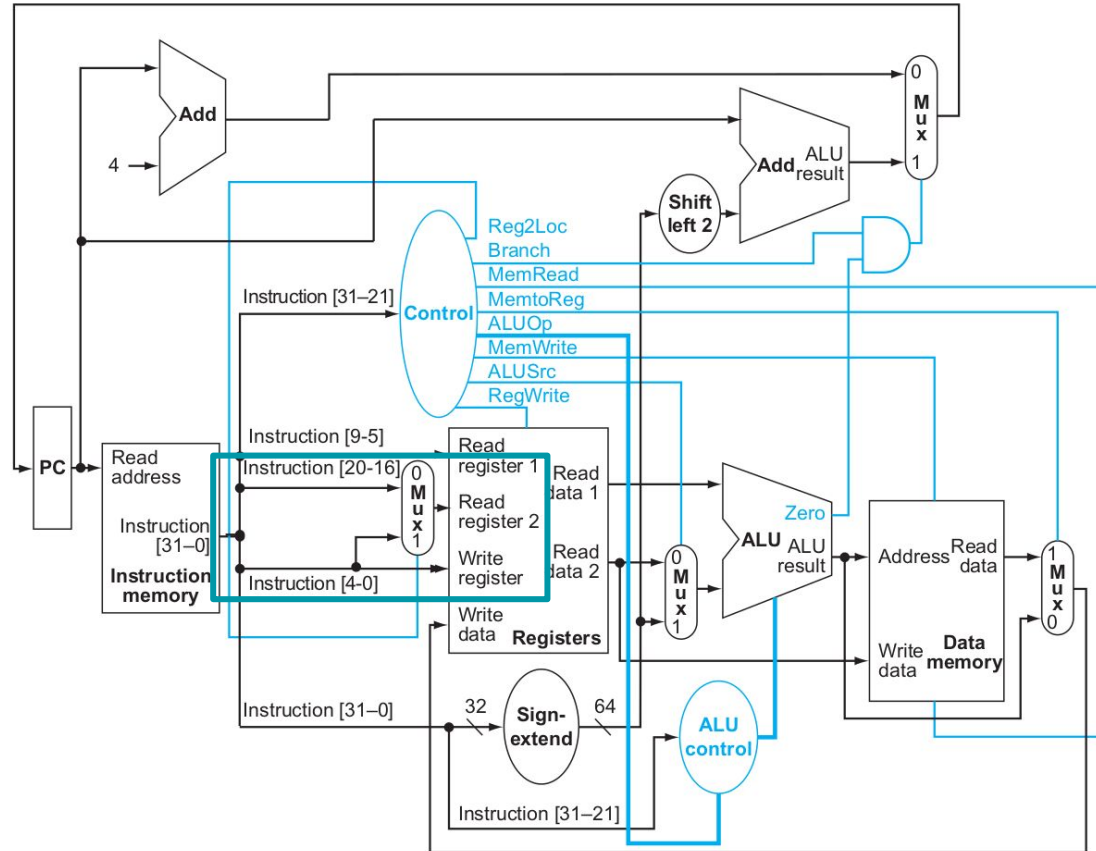
Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp
Tipo R								
LDUR								
STUR								
CBZ								





## Ejercicio 3: Reg2Loc



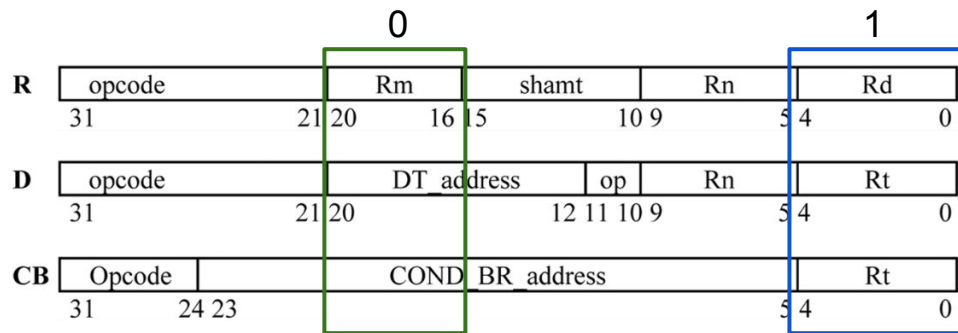
$$R[Rd] = R[Rn] + R[Rm]$$

# Ejercicio 3: Reg2Loc

Determina de qué parte de la instrucción se obtiene la dirección del registro operando 2.

1: El número de registro viene del campo Rt (bits 4:0)

0: El número de registro viene del campo Rm (bits 20:16)



$R[Rd] = R[Rn] + R[Rm]$

$R[Rt] = M[R[Rn] + DTAddr]$

$M[R[Rn] + DTAddr] = R[Rt]$

if( $R[Rt] == 0$ )

$PC = PC + CondBranchAddr$

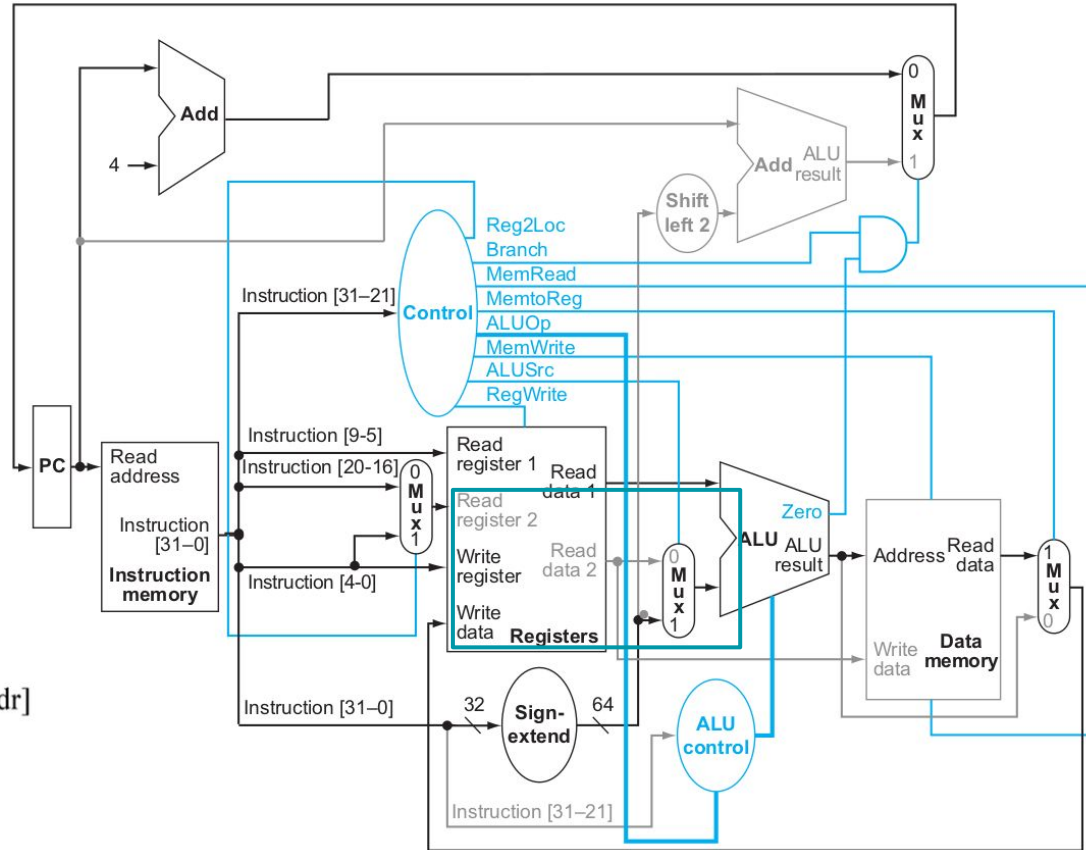
Tipo R  $\rightarrow$  Reg2Loc = 0

LDUR  $\rightarrow$  Reg2Loc = X

STUR  $\rightarrow$  Reg2Loc = 1

Tipo CB  $\rightarrow$  Reg2Loc = 1

# Ejercicio 3: Reg2Loc (LDUR)



$$R[Rt] = M[R[Rn] + DTAddr]$$

# Ejercicio 3: Reg2Loc

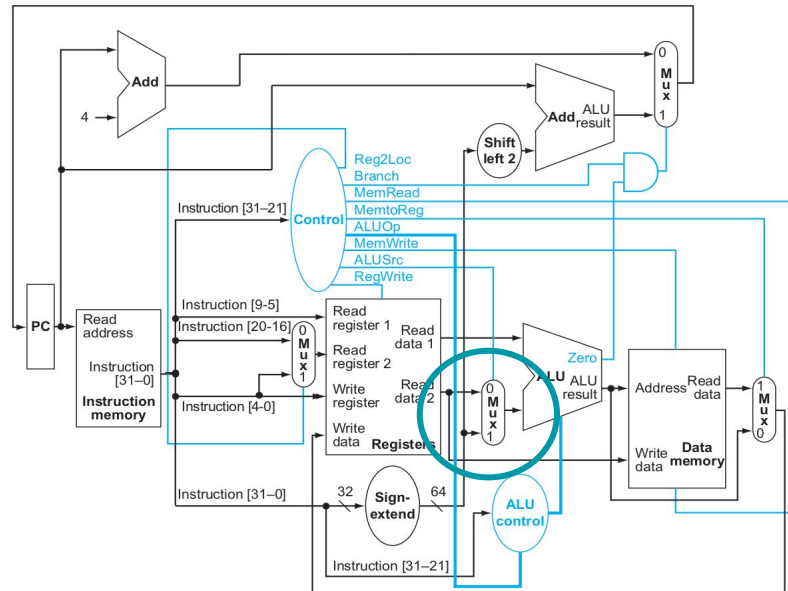
Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0							
LDUR	X							
STUR	1							
CBZ	1							

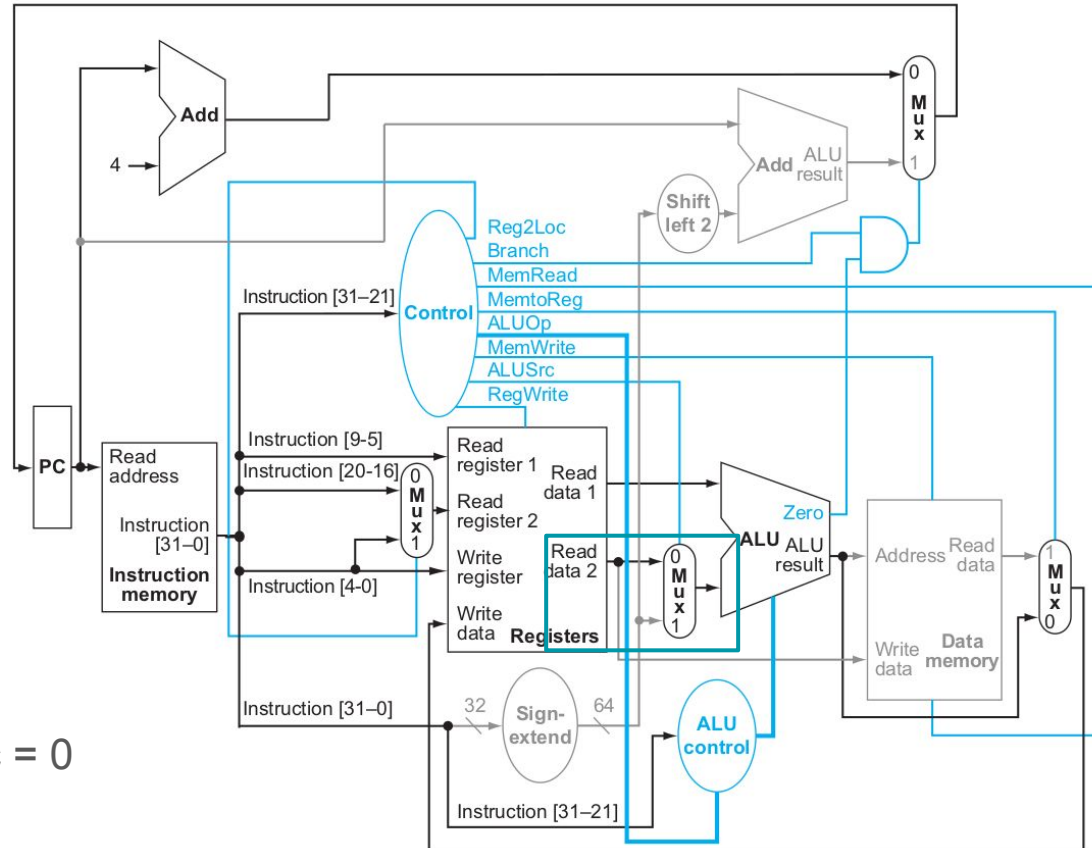
# Ejercicio 3: ALUSrc

Determina de dónde viene el segundo operando de la ALU:

- 1: Viene de la salida del bloque *sign extended*
- 0: Se obtiene de la segunda salida del *Register file*



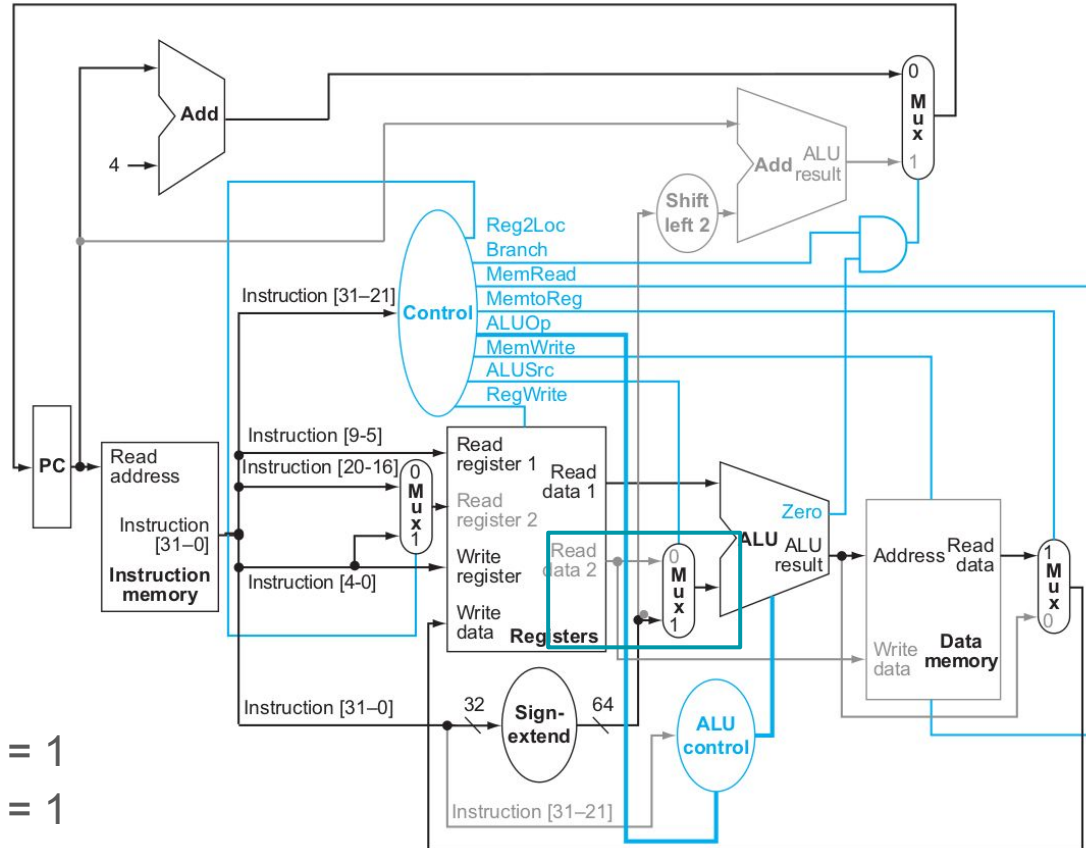
# Ejercicio 3: ALUSrc (Tipo R)



Tipo R  $\rightarrow$  ALUSrc = 0

$$R[Rd] = R[Rn] + R[Rm]$$

# Ejercicio 3: ALUSrc (LDUR)



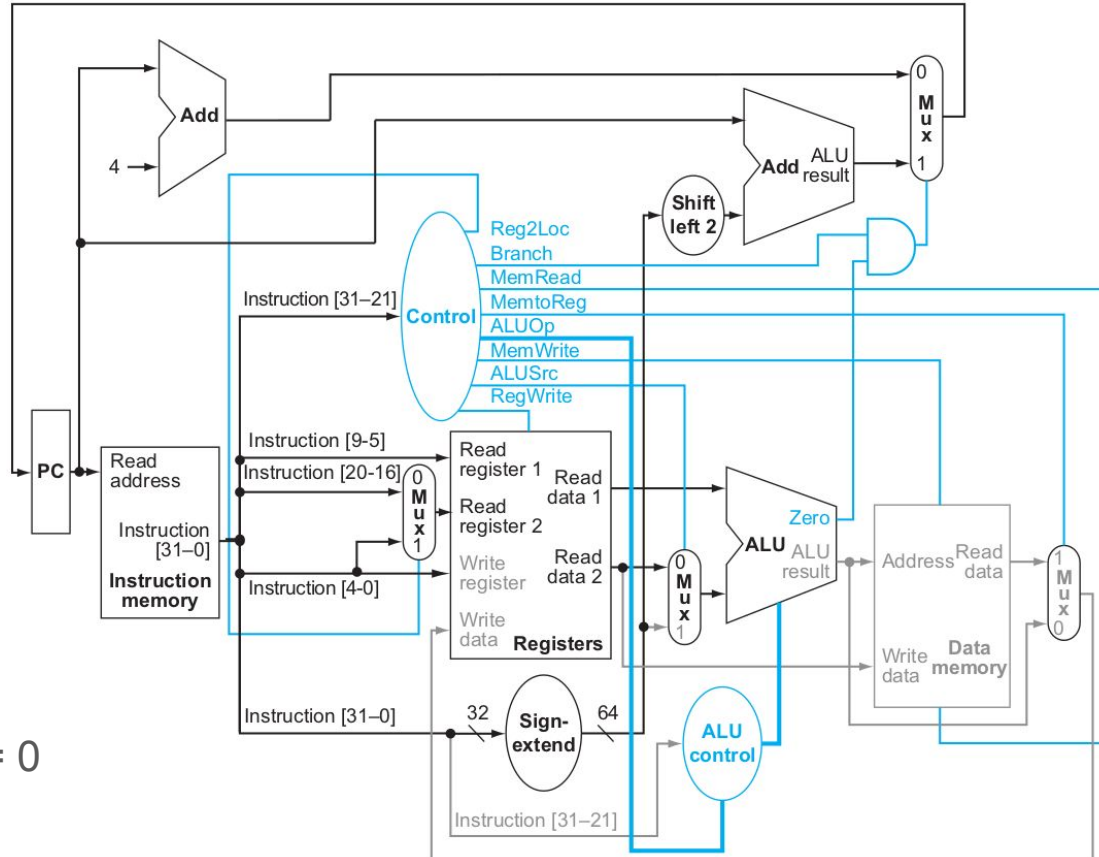
LDUR → ALUSrc = 1

STUR → ALUSrc = 0

$$R[Rt] = M[R[Rn] + DTAddr]$$

$$M[R[Rn] + DTAddr] = R[Rt]$$

# Ejercicio 3: ALUSrc (CBZ)



CBZ  $\rightarrow$  ALUSrc = 0

if(R[Rt]==0)  
PC = PC + CondBranchAddr



# Ejercicio 3: ALUSrc

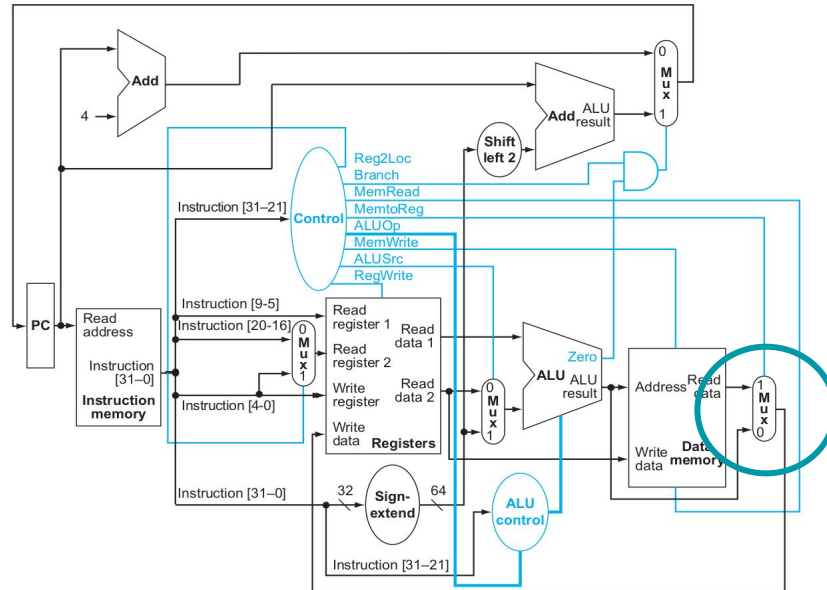
Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

Instr.	Reg2Loc	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0						
LDUR	X	1						
STUR	1	1						
CBZ	1	0						

## Ejercicio 3: MemtoReg

Determina de donde se obtiene el valor a escribirse en registro:

- 1: El dato se obtiene del *data memory*  
0: El dato se obtiene del resultado de la ALU



Tipo R  $\rightarrow$  MemtoReg = 0  
LDUR  $\rightarrow$  MemtoReg = 1  
STUR  $\rightarrow$  MemtoReg = X  
CBZ  $\rightarrow$  MemtoReg = X

$$R[Rd] = R[Rn] + R[Rm]$$

$$R[R_t] = M[R[R_n] + DTAddr]$$

$$M[R[R_n] + DTAddr] = R[R_t]$$

```
if(R[Rt]==0)
```

$$PC = PC + \text{CondBranchAddr}$$

# Ejercicio 3: MemtoReg

Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

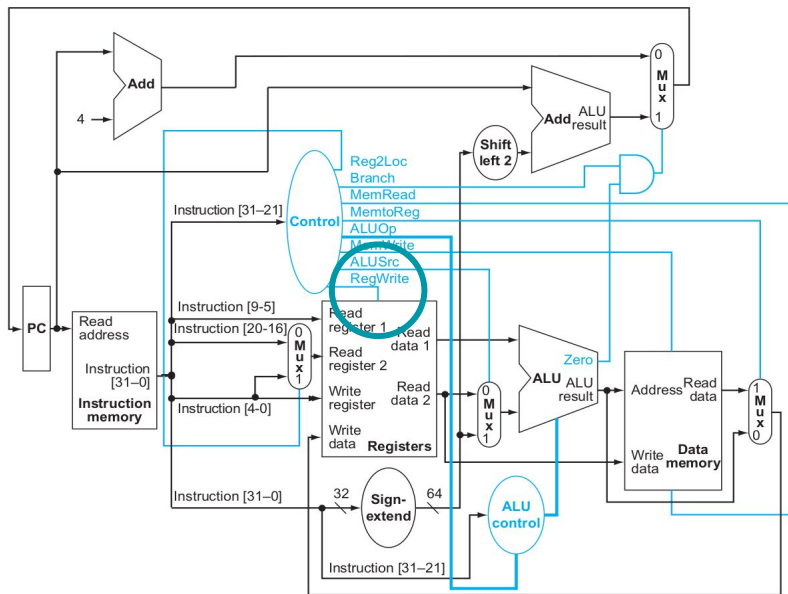
Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0					
LDUR	X	1	1					
STUR	1	1	X					
CBZ	1	0	X					

## Ejercicio 3: RegWrite

## Determina si un dato debe escribirse en registro:

1: El dato (Write Data) se escribe en el registro (Write Register)

0: No se escribe dato alguno.



Tipo R  $\rightarrow$  RegWrite = 1

LDUR  $\rightarrow$  RegWrite = 1

STUR  $\rightarrow$  RegWrite = 0

CBZ  $\rightarrow$  RegWrite = 0

$$R[Rd] = R[Rn] + R[Rm]$$

$$R[Rt] = M[R[Rn] + DTAddr]$$

$$M[R[R_n] + DTAddr] = R[R_t]$$

```
if(R[Rt]==0)
```

$$PC = PC + \text{CondBranchAddr}$$

# Ejercicio 3: RegWrite

Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

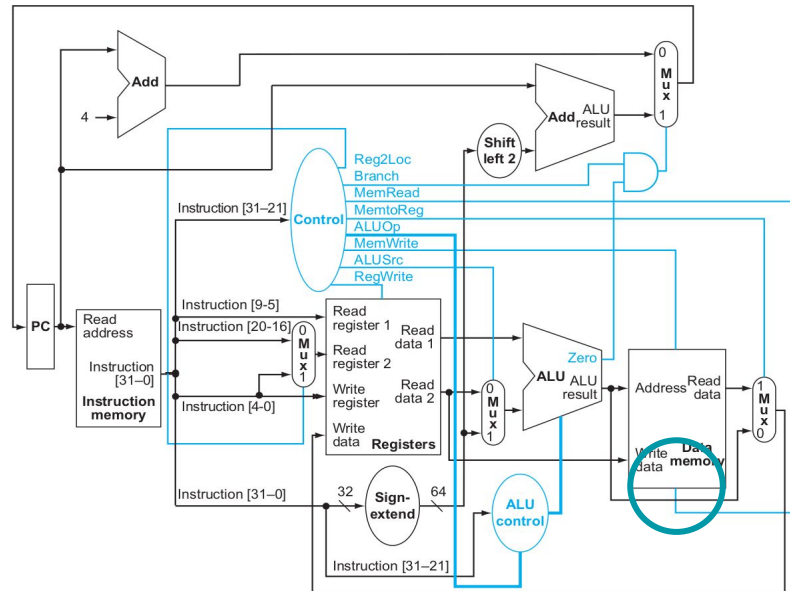
Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1				
LDUR	X	1	1	1				
STUR	1	1	X	0				
CBZ	1	0	X	0				

# Ejercicio 3: MemRead

Determina si un dato debe leerse de la *Data Memory*:

- 1: En la salida *Read data* se obtiene el dato apuntado por *address*
- 0: No se lee dato alguno.

Tipo R  $\rightarrow$  RegWrite = 0  
LDUR  $\rightarrow$  RegWrite = 1  
STUR  $\rightarrow$  RegWrite = 0  
CBZ  $\rightarrow$  RegWrite = 0



$$R[Rd] = R[Rn] + R[Rm]$$

$$R[Rt] = M[R[Rn] + DTAddr]$$

$$M[R[Rn] + DTAddr] = R[Rt]$$

$$\text{if}(R[Rt] == 0)$$

$$PC = PC + \text{CondBranchAddr}$$

# Ejercicio 3: MemRead

Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

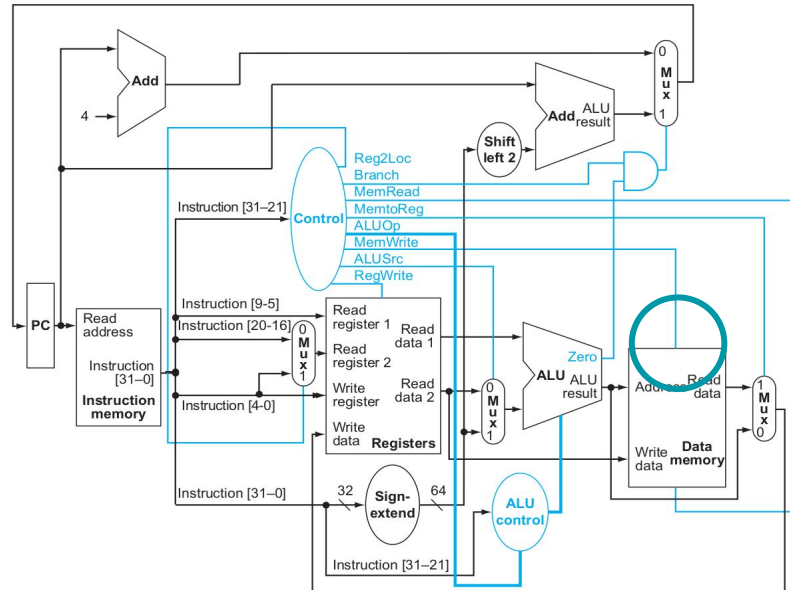
Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1	0			
LDUR	X	1	1	1	1			
STUR	1	1	X	0	0			
CBZ	1	0	X	0	0			

# Ejercicio 3: MemWrite

Determina si un dato debe escribirse en la *Data Memory*:

- 1: Se escribe el dato (*Write data*) en la dirección apuntada por *address*
- 0: No se escribe dato alguno.

Tipo R  $\rightarrow$  RegWrite = 0  
LDUR  $\rightarrow$  RegWrite = 0  
STUR  $\rightarrow$  RegWrite = 1  
CBZ  $\rightarrow$  RegWrite = 0



$$R[Rd] = R[Rn] + R[Rm]$$

$$R[Rt] = M[R[Rn] + DTAddr]$$

$$M[R[Rn] + DTAddr] = R[Rt]$$

if( $R[Rt] == 0$ )

$$PC = PC + CondBranchAddr$$



# Ejercicio 3: MemWrite

Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1	0	0		
LDUR	X	1	1	1	1	0		
STUR	1	1	X	0	0	1		
CBZ	1	0	X	0	0	0		

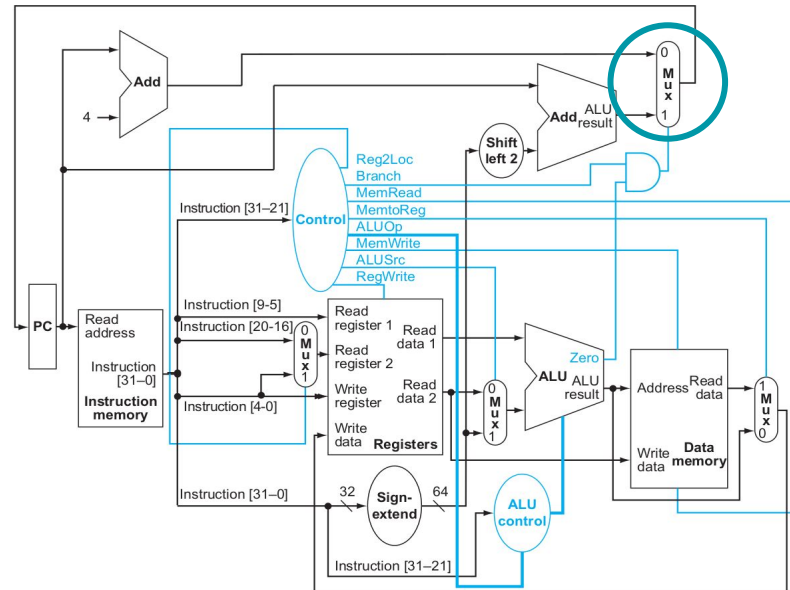
# Ejercicio 3: Branch

Determina que se está ejecutando una instrucción de salto:

1: Si la bandera Zero es 1, se reemplaza el valor de PC por la salida del sumador que computa la dirección de salto

0: PC se reemplaza por PC+4.

Tipo R  $\rightarrow$  Branch = 0  
LDUR  $\rightarrow$  Branch = 0  
STUR  $\rightarrow$  Branch = 0  
CBZ  $\rightarrow$  Branch = 1



$$R[Rd] = R[Rn] + R[Rm]$$

$$R[Rt] = M[R[Rn] + DTAddr]$$

$$M[R[Rn] + DTAddr] = R[Rt]$$

$$\text{if}(R[Rt] == 0)$$

$$PC = PC + \text{CondBranchAddr}$$

# Ejercicio 3: Branch

Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1	0	0	0	
LDUR	X	1	1	1	1	0	0	
STUR	1	1	X	0	0	1	0	
CBZ	1	0	X	0	0	0	1	

# Ejercicio 3: ALUop

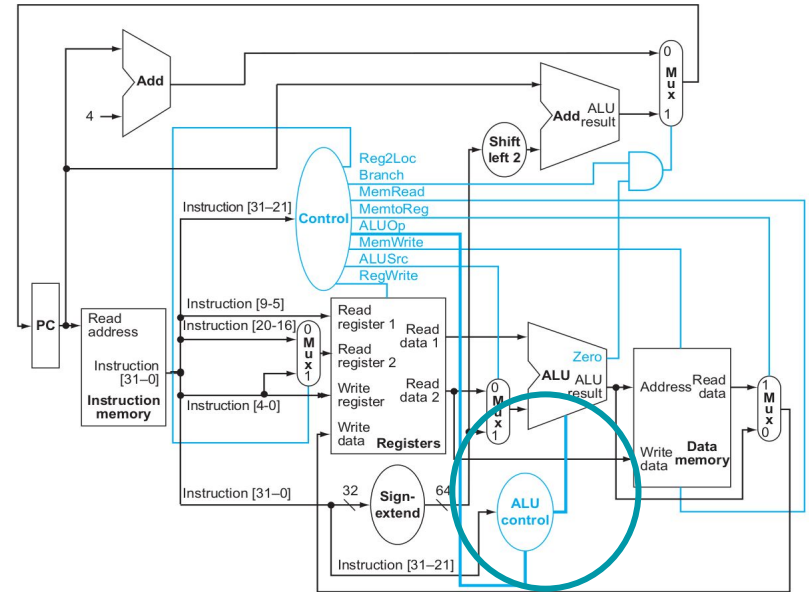
Determina qué operación debe realizar la ALU. Esta señal de control es de 2 bits

00 → Add

01 → pass input b

10 → Se determina con el opcode:

- ADD
- SUB
- AND
- OR



# Ejercicio 3: ALUop

Instruction	ALUOp	Instruction operation	Opcode field	Desired ALU action	ALU control input
LDUR	00	load register	XXXXXXXXXXXX	add	0010
STUR	00	store register	XXXXXXXXXXXX	add	0010
CBZ	01	compare and branch on zero	XXXXXXXXXXXX	pass input b	0111
R-type	10	ADD	10001011000	add	0010
R-type	10	SUB	11001011000	subtract	0110
R-type	10	AND	10001010000	AND	0000
R-type	10	ORR	10101010000	OR	0001

$R[Rt] = M[R[Rn] + DTAddr]$

$M[R[Rn] + DTAddr] = R[Rt]$

if( $R[Rt] == 0$ )

$PC = PC + CondBranchAddr$

$R[Rd] = R[Rn] + R[Rm]$

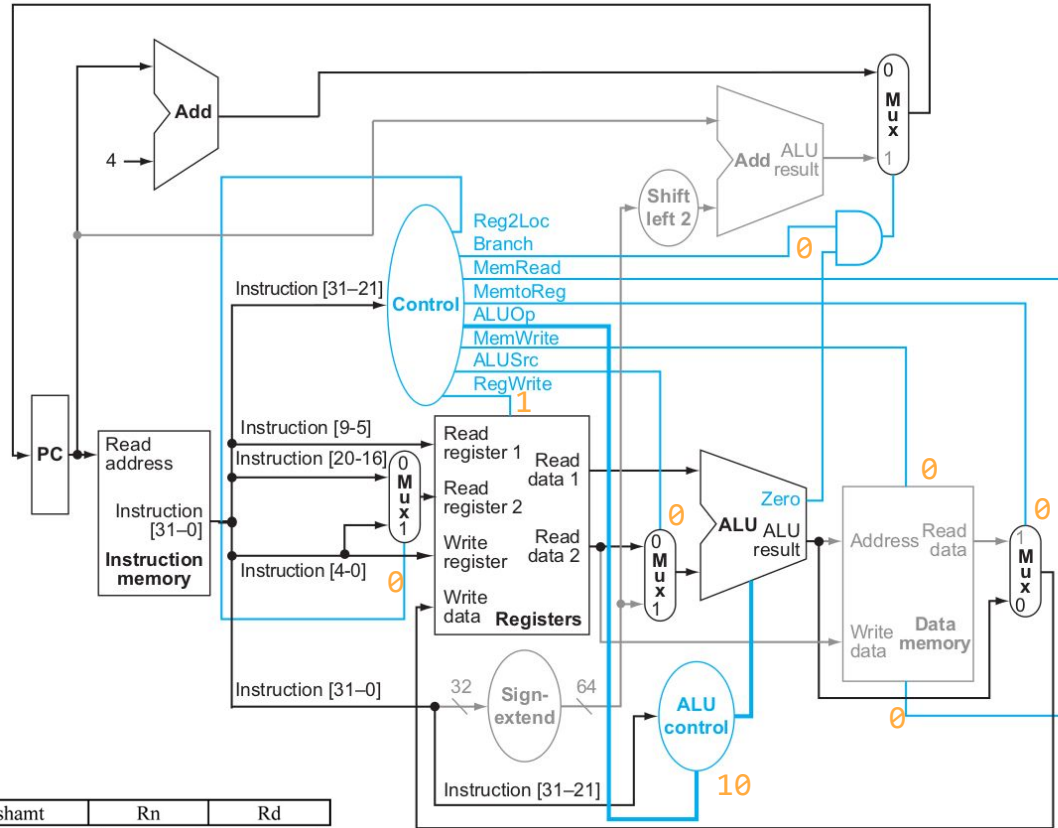
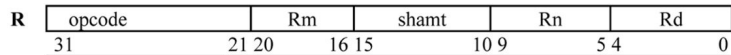
# Ejercicio 3: ALUop

Complete la tabla con el estado de las señales **sin mirar el libro**. Indique con X las condiciones no-importa.

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1	0	0	0	10
LDUR	X	1	1	1	1	0	0	00
STUR	1	1	X	0	0	1	0	00
CBZ	1	0	X	0	0	0	1	01

# Tipo R

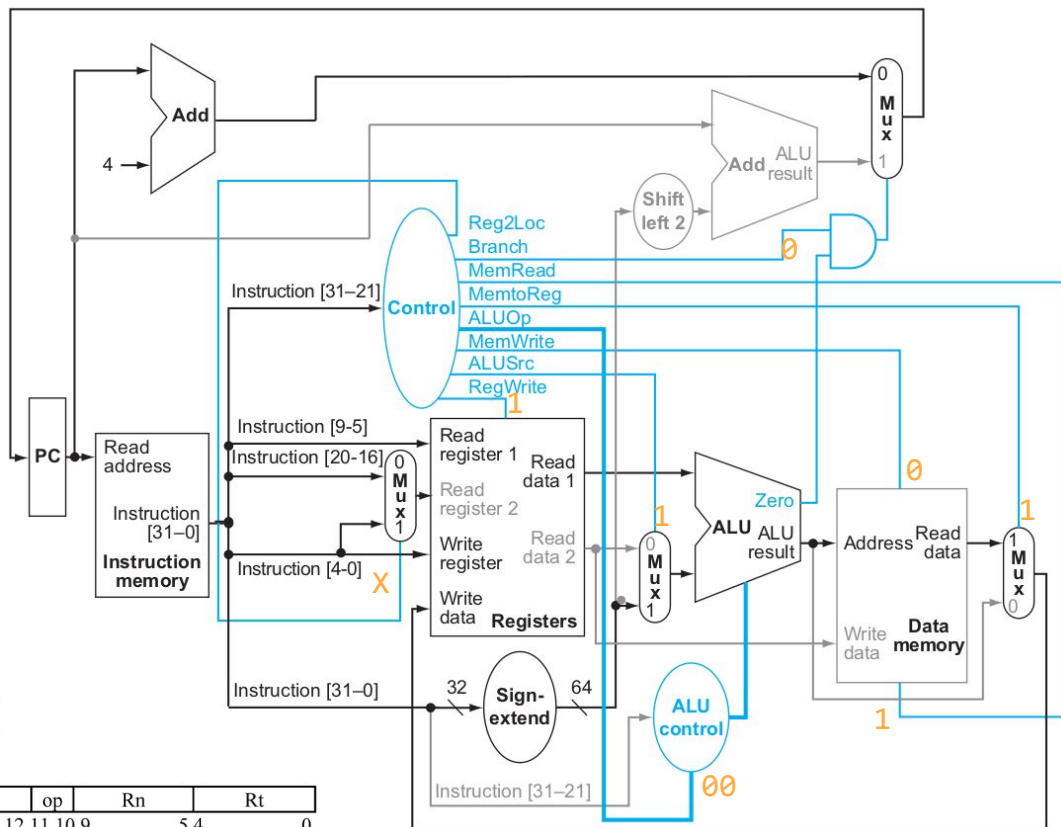
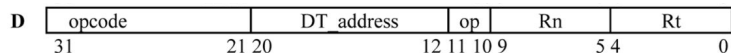
$$R[R_d] = R[R_n] + R[R_m]$$



Instr.	Reg2Loc	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp
Tipo R	0	0	0	1	0	0	0	10

# LDUR

$$R[Rt] = M[R[Rn] + DTAddr]$$

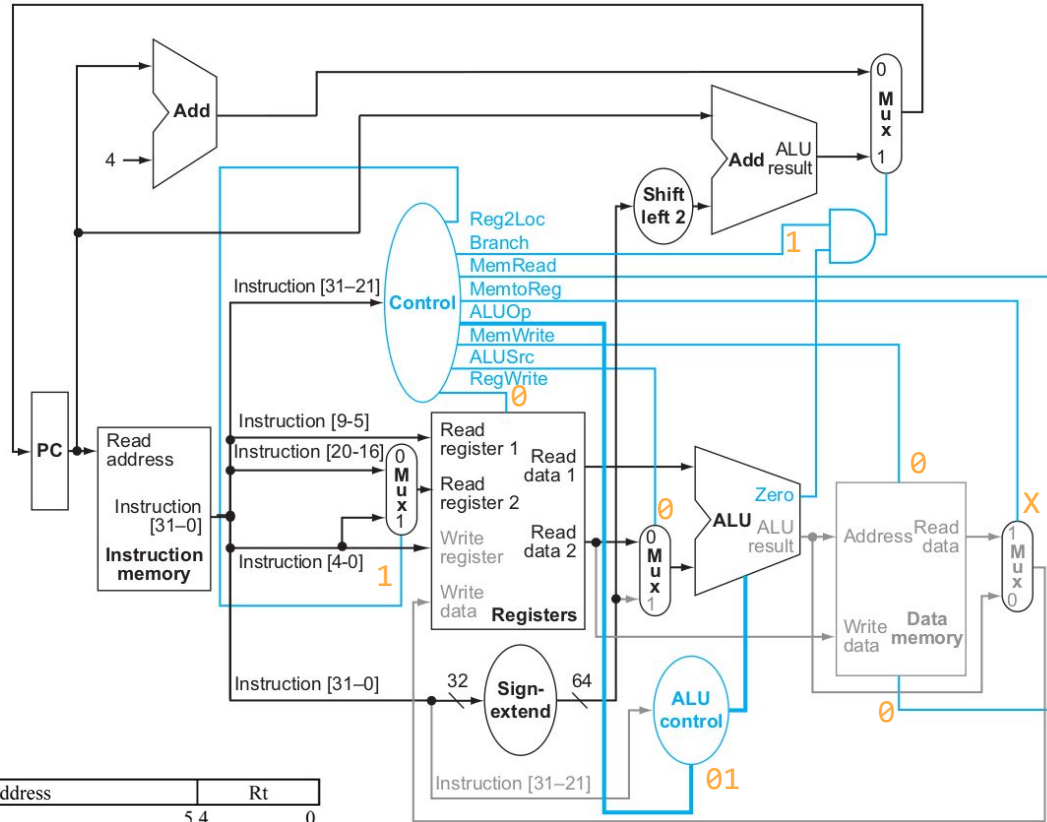


Instr.	Reg2Loc	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp
LDUR	X	1	1	1	1	0	0	00



# CBZ

if(R[Rt]==0)  
PC = PC + CondBranchAddr



Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp
CBZ	1	0	X	0	0	0	1	01

# Ejercicio 4

Cuando se fabrican los chips de silicio, defectos en los materiales y errores en la fabricación pueden generar circuitos defectuosos. Un defecto común es que un cable de señal se rompa y siempre registre un '0' lógico. Esto se conoce comúnmente como "[\*stuck-at-0 fault\*](#)".

- 4.1) ¿Qué instrucciones operarían de forma incorrecta si el cable MemToReg está atascado en '0'?
- 4.2) ¿Qué instrucciones operarían de forma incorrecta si el cable ALUSrc está atascado en '0'?
- 4.3) ¿Qué instrucciones operarían de forma incorrecta si el cable Reg2Loc está atascado en '0'?

# Ejercicio 4

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1	0	0	0	10
LDUR	X	1	1	1	1	0	0	00
STUR	1	1	X	0	0	1	0	00
CBZ	1	0	X	0	0	0	1	01

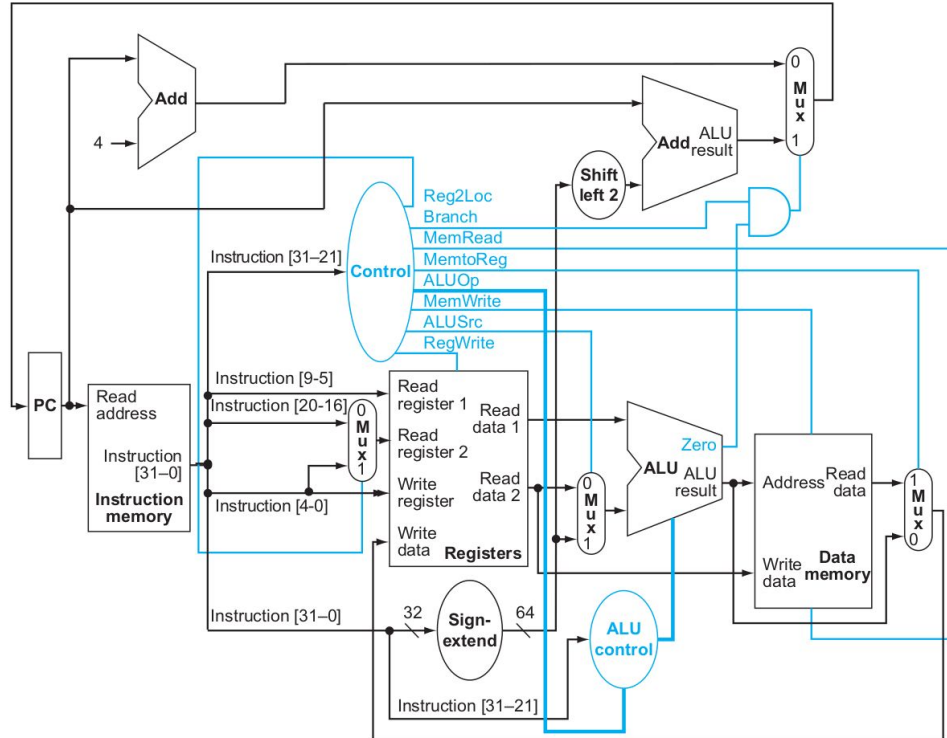
## Ejercicio 4: Rta

Instr.	Reg2Loc	ALUsrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUop
Tipo R	0	0	0	1	0	0	0	10
LDUR	X	1	1	1	1	0	0	00
STUR	1	1	X	0	0	1	0	00
CBZ	1	0	X	0	0	0	1	01
	4.3	4.2	4.1					

# Ejercicio 5

Agregando una compuerta en diagrama del *data path & control*, cambie la implementación de la instrucción CBZ a CBNZ.

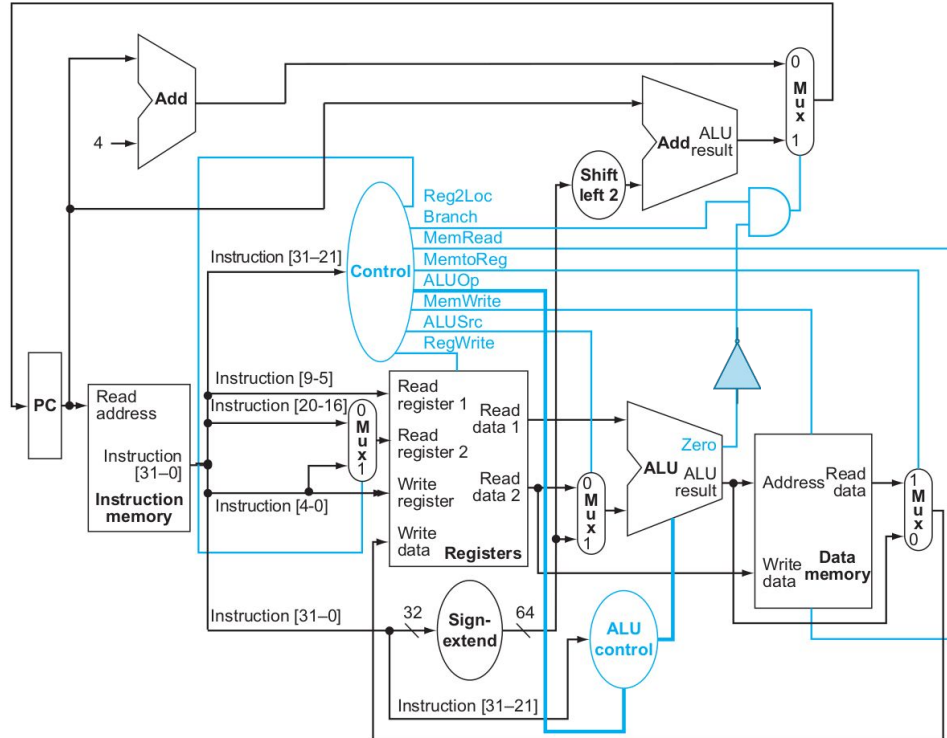
Microprocesador  
con CBZ



# Ejercicio 5: Rta

Agregando una compuerta en diagrama del *data path & control*, cambie la implementación de la instrucción CBZ a CBNZ.

Microprocesador  
con CBNZ



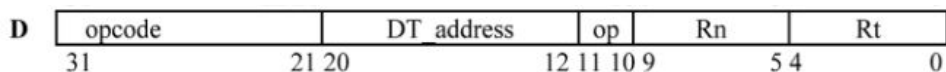
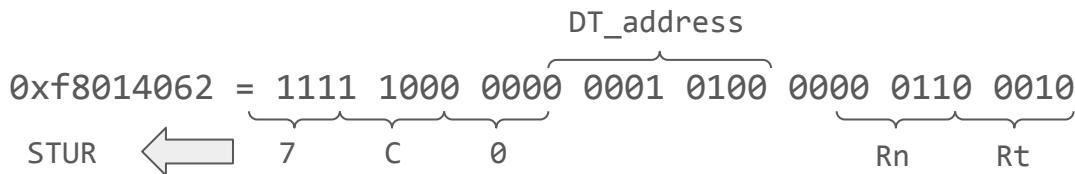
# Ejercicio 6

En este ejercicio analizaremos en detalle cómo se ejecuta una instrucción en el *single-cycle datapath*, asumiendo que la palabra de instrucción que ejecuta el procesador es: 0xf8014062, dado que PC=0x100.

- 6.1) ¿Cuáles son las salidas de los bloques Sign-extend y Shift left 2 para esta palabra de instrucción?
- 6.2) ¿Cuáles son los valores de entrada a la unidad ALU control para esta palabra de instrucción?
- 6.3) ¿Cuál es la nueva dirección en el PC después de ejecutar esta instrucción?
- 6.4) Mostrar los valores de las entradas y salidas de cada Mux durante la ejecución de esta instrucción. Para los valores que son salidas de Registers, utilizar “Reg [Xn]”.
- 6.5) ¿Cuáles son los valores de entrada de la ALU y las dos unidades Add?
- 6.6) ¿Cuáles son los valores de todas las entradas del bloque Registers?

# Ejercicio 6

En este ejercicio analizaremos en detalle cómo se ejecuta una instrucción en el *single-cycle datapath*, asumiendo que la palabra de instrucción que ejecuta el procesador es:  $0xf8014062$ , dado que  $PC=0x100$ .



$$M[R[Rn] + DTAddr] = R[Rt]$$

DT\_address =  $0b000010100$

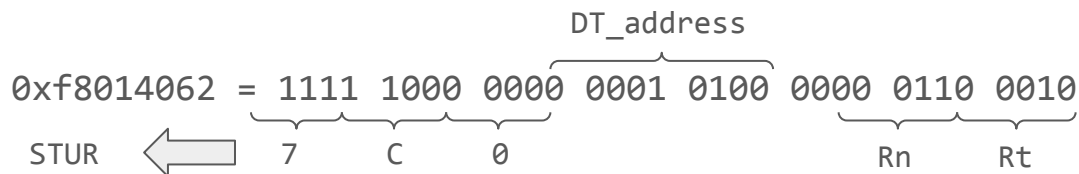
Rn =  $0b00011 = 3$

Rt =  $0b00010 = 2$



# Ejercicio 6.1

¿Cuáles son las salidas de los bloques Sign-extend y Shift left 2 para esta palabra de instrucción?  
sign-extend



DT\_address = 0b000010100

DTAddr = 0x00000000000000014

$M[R[Rn] + DTAddr] = R[Rt]$

$DTAddr = \{ 55\{DT\_address[8]\}, DT\_address \}$

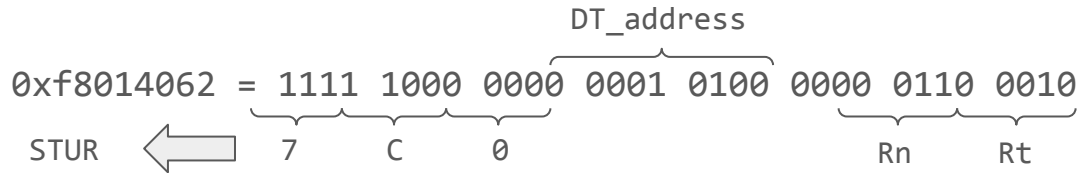
shift left 2

DTAddr = 0x00000000000000014

DTAddr << 2 = 0x00000000000000050

# Ejercicio 6.2

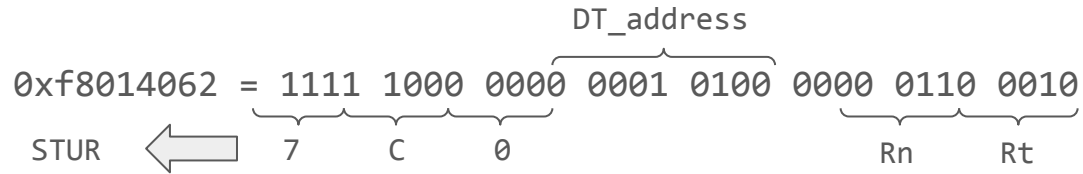
¿Cuáles son los valores de entrada a la unidad ALU control para esta palabra de instrucción?



Instruction	ALUOp	Instruction operation	Opcode field	Desired ALU action	ALU control input
LDUR	00	load register	XXXXXXXXXX	add	0010
STUR	00	store register	XXXXXXXXXX	add	0010
CBZ	01	compare and branch on zero	XXXXXXXXXX	pass input b	0111
R-type	10	ADD	10001011000	add	0010
R-type	10	SUB	11001011000	subtract	0110
R-type	10	AND	10001010000	AND	0000
R-type	10	ORR	10101010000	OR	0001

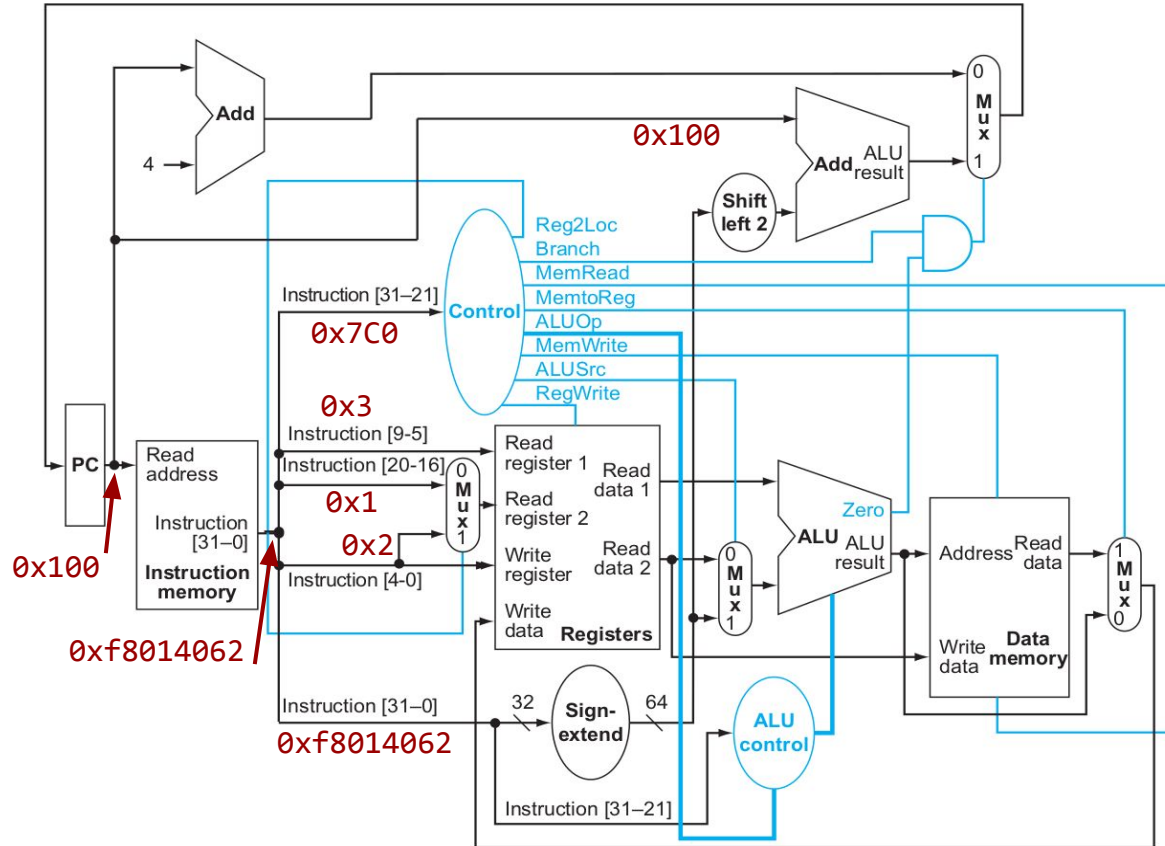
## Ejercicio 6.3

¿Cuál es la nueva dirección en el PC después de ejecutar esta instrucción?

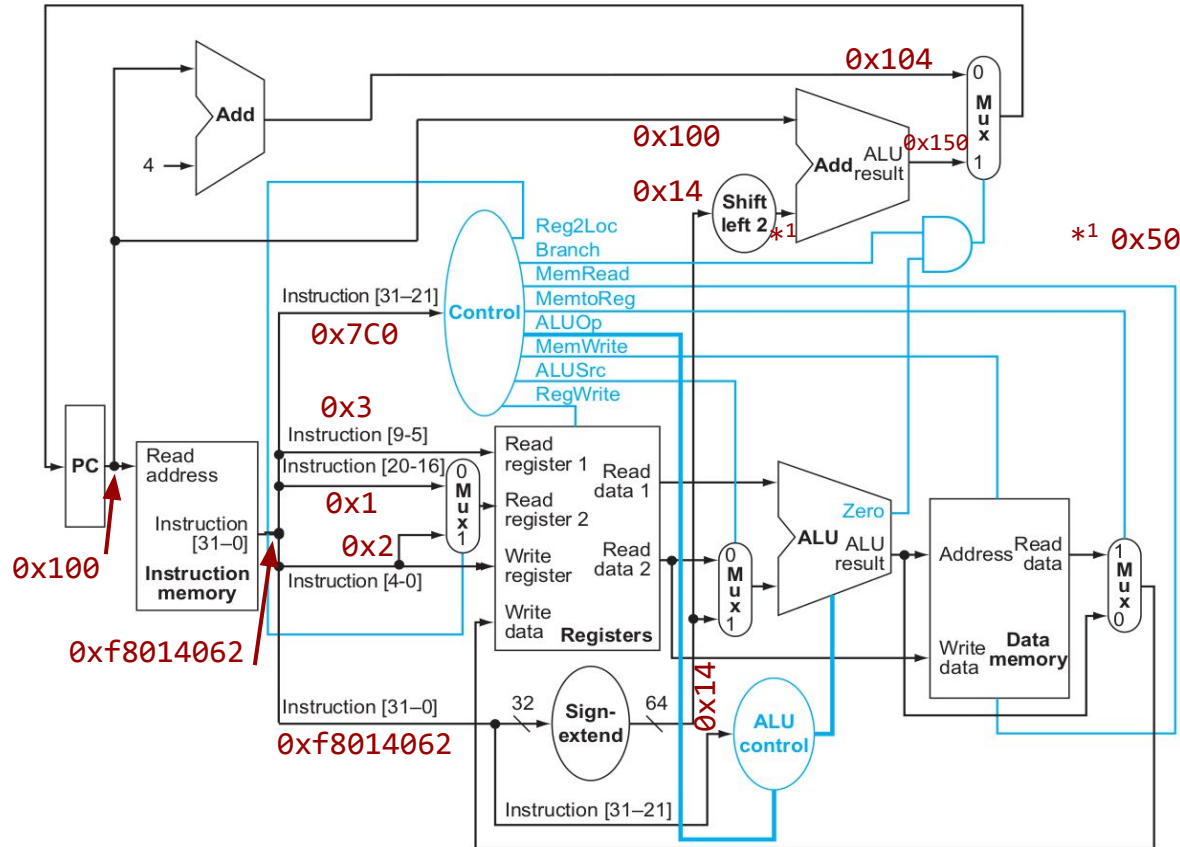


$$PC = PC + 4 = 0x100 + 0x004 = 0x104$$

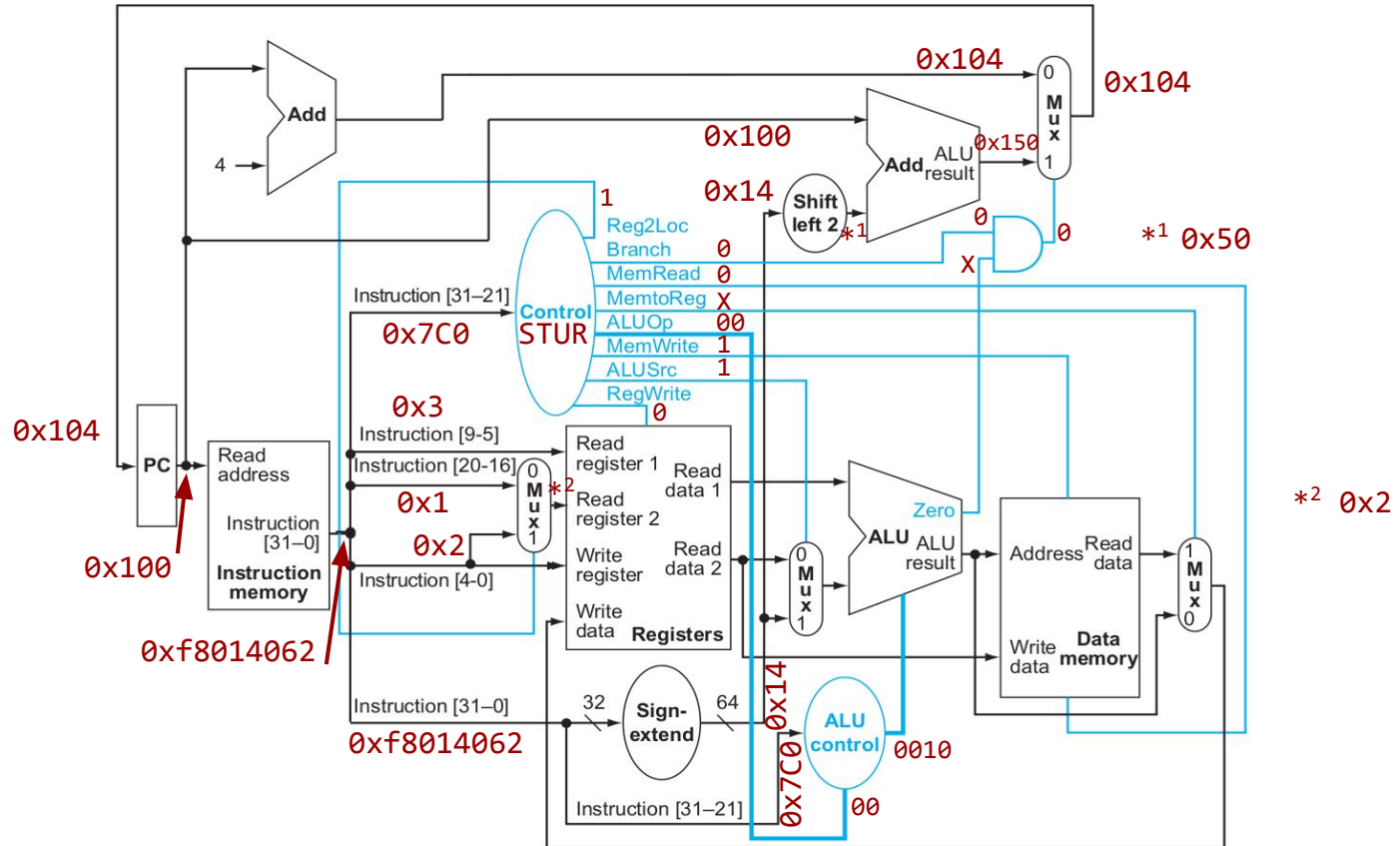
# Ejercicio 6.4, 6.5 y 6.6



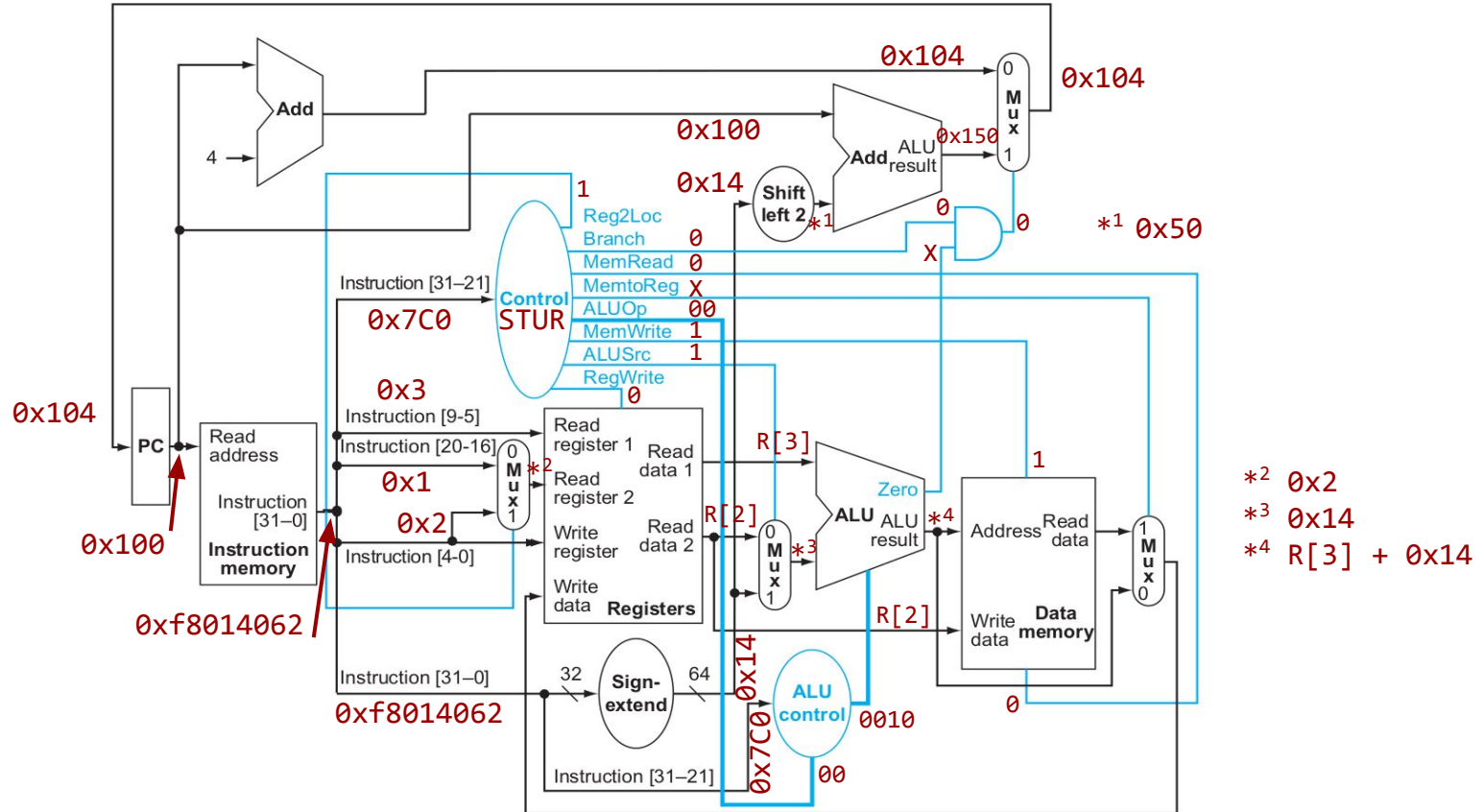
# Ejercicio 6.4, 6.5 y 6.6



# Ejercicio 6.4, 6.5 y 6.6

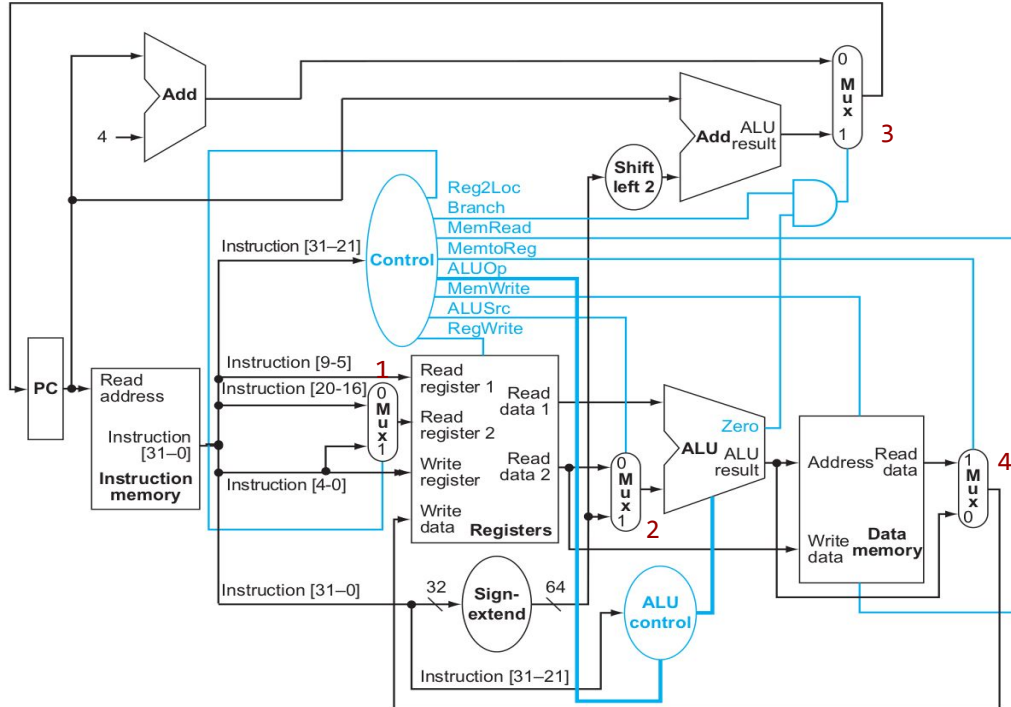


# Ejercicio 6.4, 6.5 y 6.6



# Ejercicio 6.4 (Rta)

Mostrar los valores de las entradas y salidas de cada Mux durante la ejecución de esta instrucción. Para los valores que son salidas de Registers, utilizar “Reg [Xn]”.

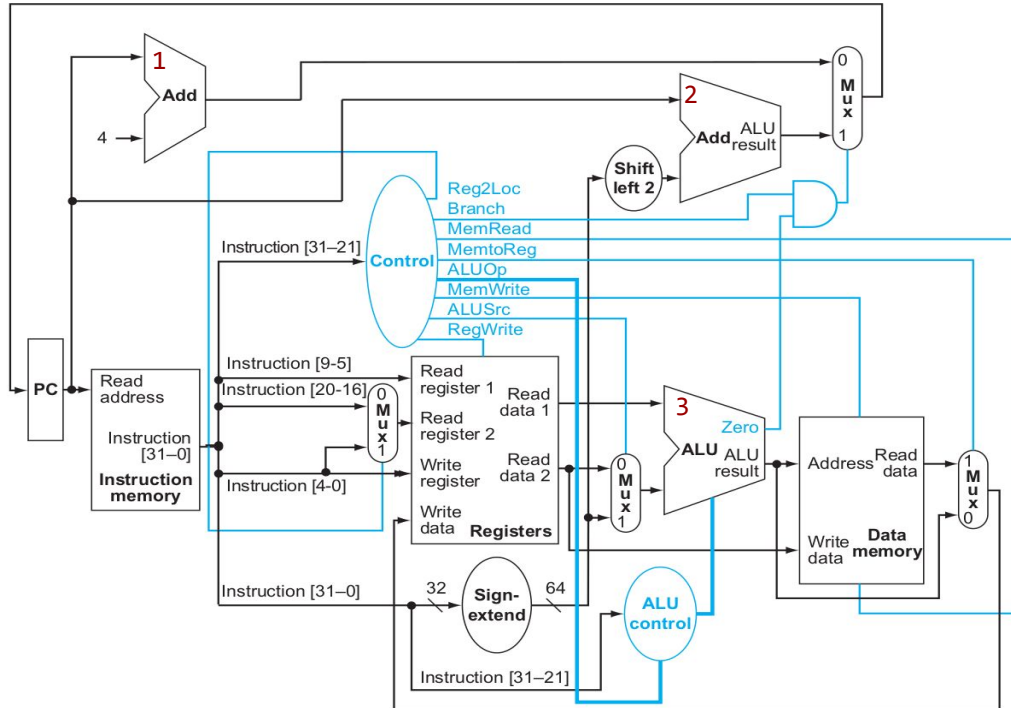


Mux	Valor de Salida
1	0x2
2	0x14
3	0x104
4	X



# Ejercicio 6.5 (Rta)

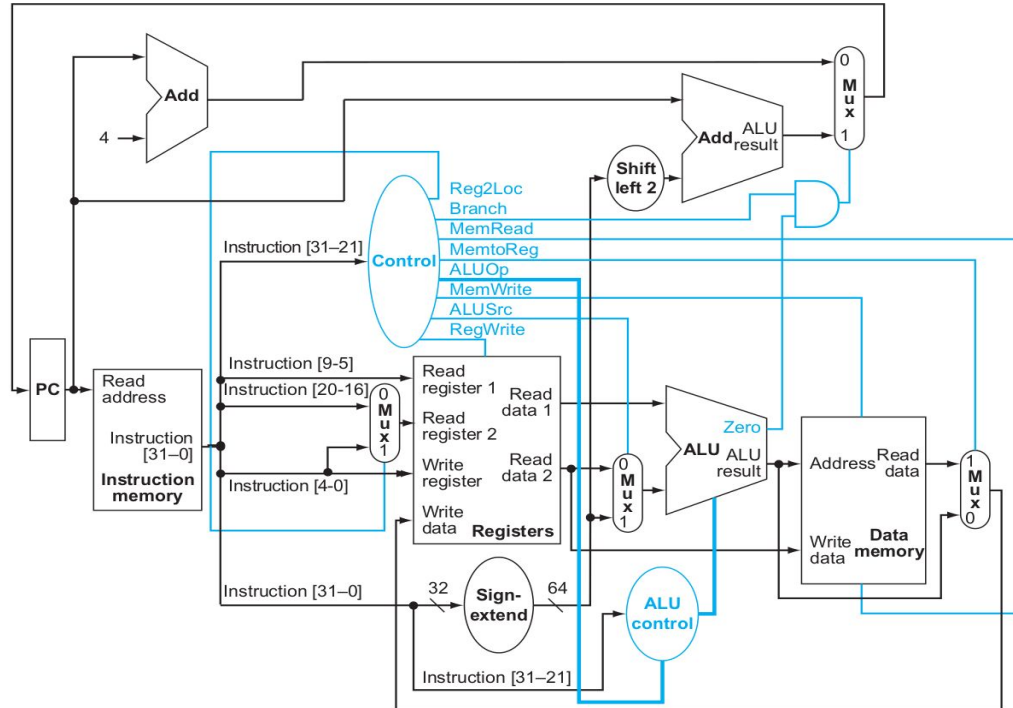
¿Cuáles son los valores de entrada de la ALU y las dos unidades Add?



Modulo	Valor de Salida
1	0x104
2	0x150
3	R[3] + 0x14

# Ejercicio 6.6 (Rta)

¿Cuáles son los valores de todas las entradas del bloque Registers?



Entrada	Valor
Read Register 1	0x3
Read Register 2	0x2
Write Register	0x2
Write Data	X