TDC_4. Diseño de un procesador (VHDL)

UCA
Universidad de Cádiz

Repaso de:

- Elementos constructivos de un computador simple.
- Repaso del diseño de un repertorio de instrucciones (ISA)
- Implementación del ISA: Diferentes microarquitecturas

UCA
Universidad de Cádiz

Objetivos:

- Descripción VHDL de los principales elementos de un computador.
- Diseño de un procesador simple mediante VHDL
- Verificación del diseño en placa desarrollo

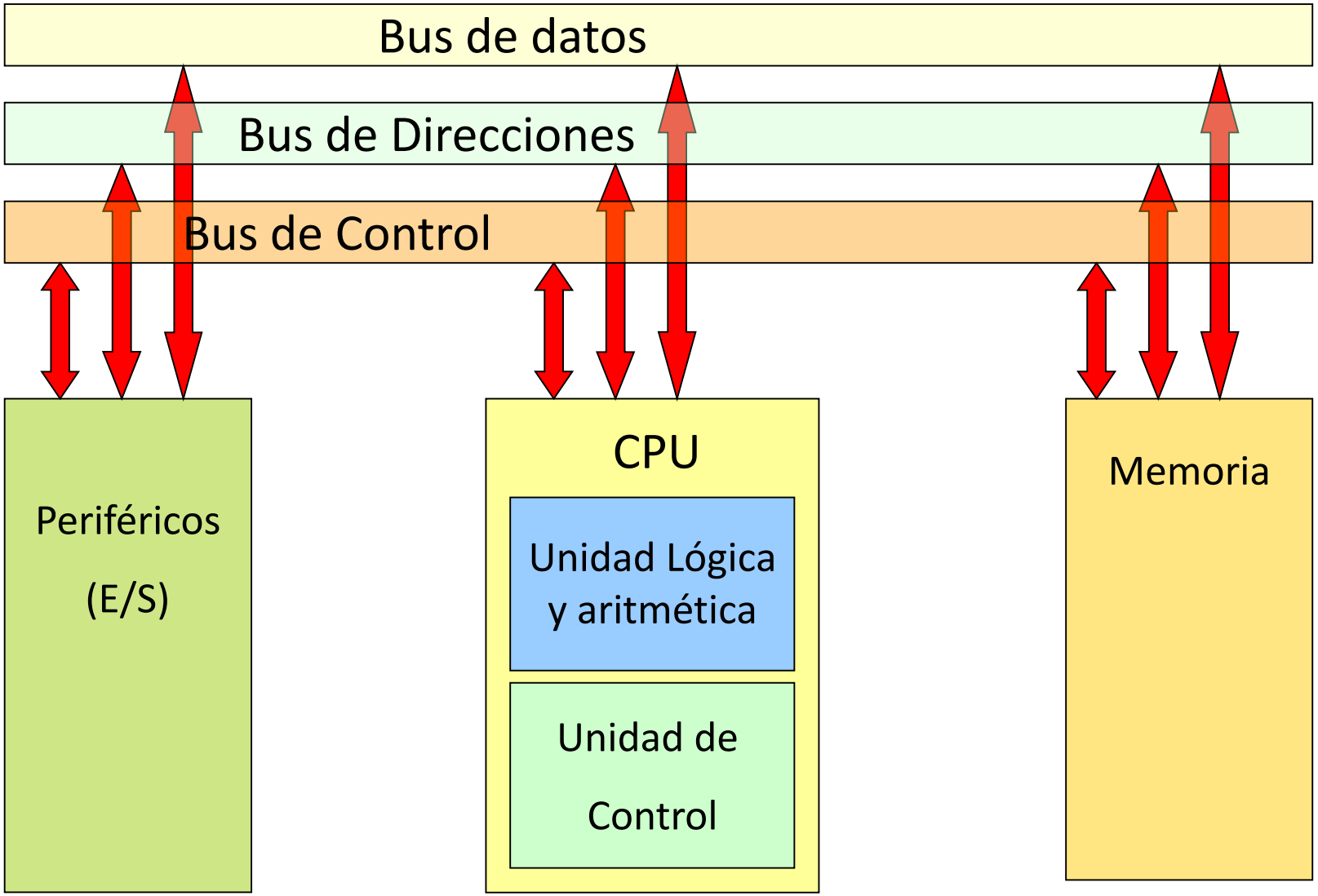
UCA
Universidad de Cádiz

4.1. Elementos de un computador simple

UCA
Universidad de Cádiz

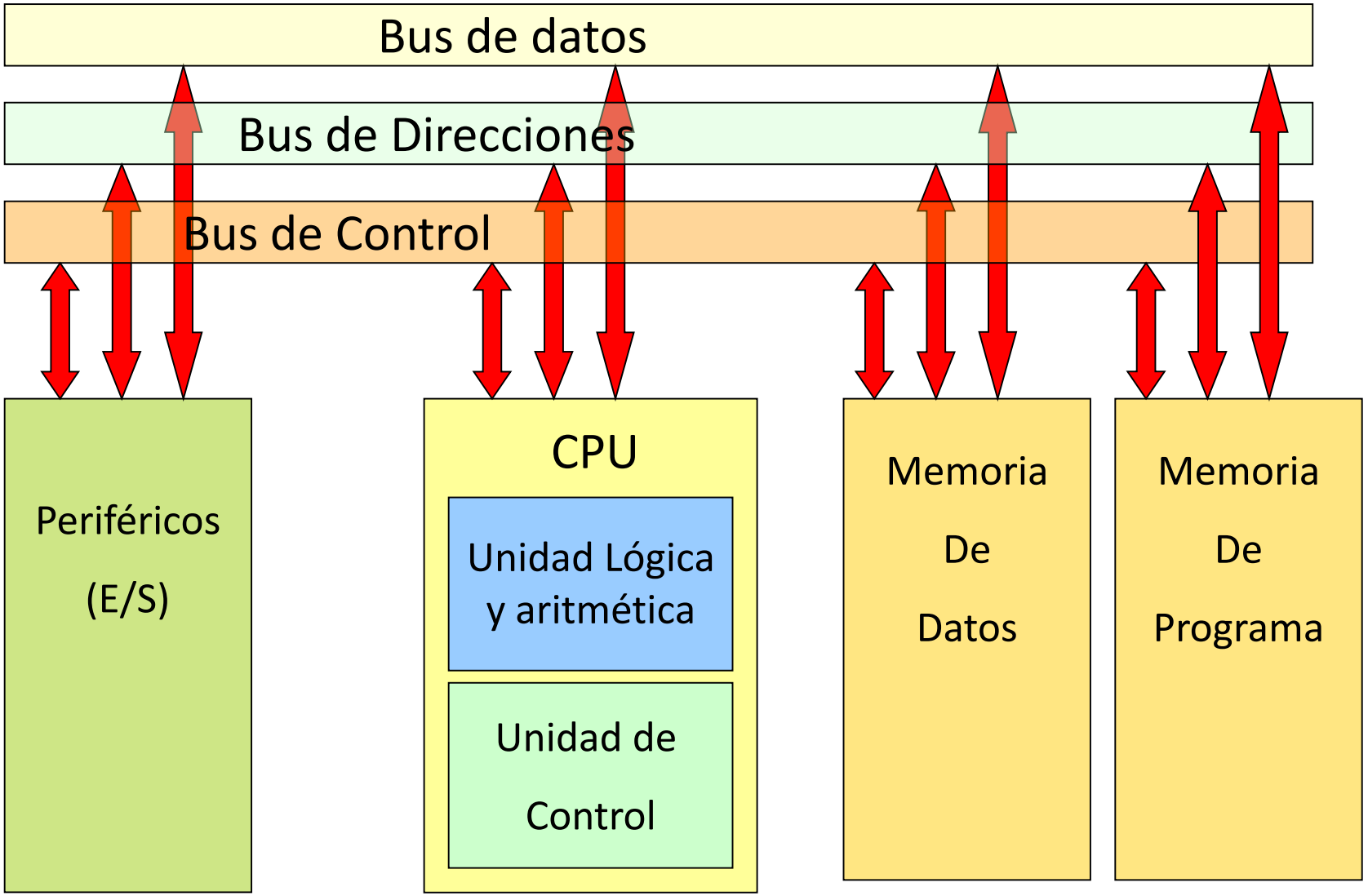
Elementos de un computador simple

Arquitectura Von Neumann



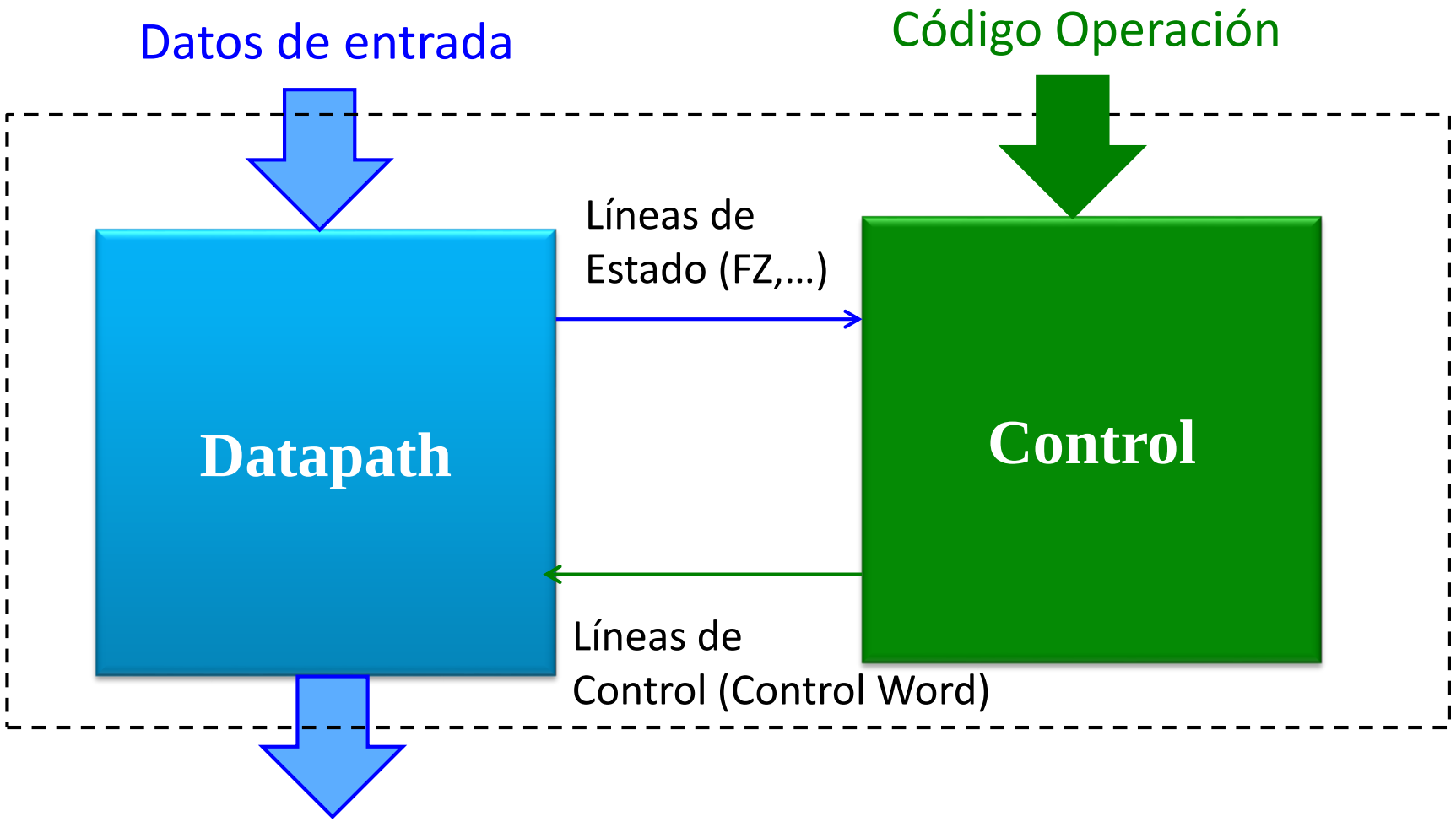
Elementos de un computador simple

Arquitectura Harvard



Elementos de un computador simple

Unidad central de proceso (CPU)



4.2. Camino de datos (DataPath)

UCA
Universidad de Cádiz

Camino de datos (Datapath)

- Elementos encargados de manipular los datos (información)
- Consta de tres tipos de elementos:
  - (1) Unidadades funcionales: ALU, desplazadores, etc
  - (2) Elementos de almacenamiento: registros
  - (3) Buses y multiplexores
- Su funcionamiento es gobernado por las líneas de control procedentes de la **unidad de control**.
- Avisa de sucesos especiales sucedidos durante las operaciones (cero, signo, desbordamiento, etc).
- Implementa el ISA establecido (varias uArquitecturas).
- **Tipos :**
  - ✓ **General**
  - ✓ **Dedicado (p.e.: solo suma)**

Tipos de Camino de datos

- **Clasificación según forma de almacenar datos establecido en ISA**
  - **Acumulador**
  - **Memoria-Memoria**
  - **Memoria-Registro**
  - **Registro-Registro (Load-Store)**
  - **Pila**

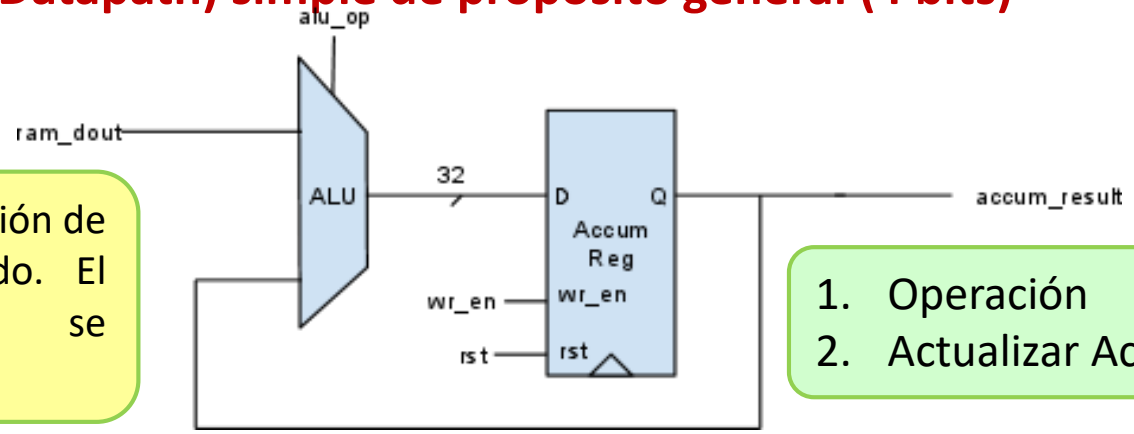
C = A + B

Acumulador	Memoria-Memoria	Registro-Memoria	Pila	Registro (load-store)
			`Push A`	`Load r1,A`
`Load A`	`Add C,B,A`	`Load r1,A`	`Push B`	`Load r2,B`
`Add B`		`Add r1,B`	`Add`	`Add r3,r1,r2`
`Store C`		`Store C,r1`	`Pop C`	`Store C,r3`

Tipos de Camino de datos

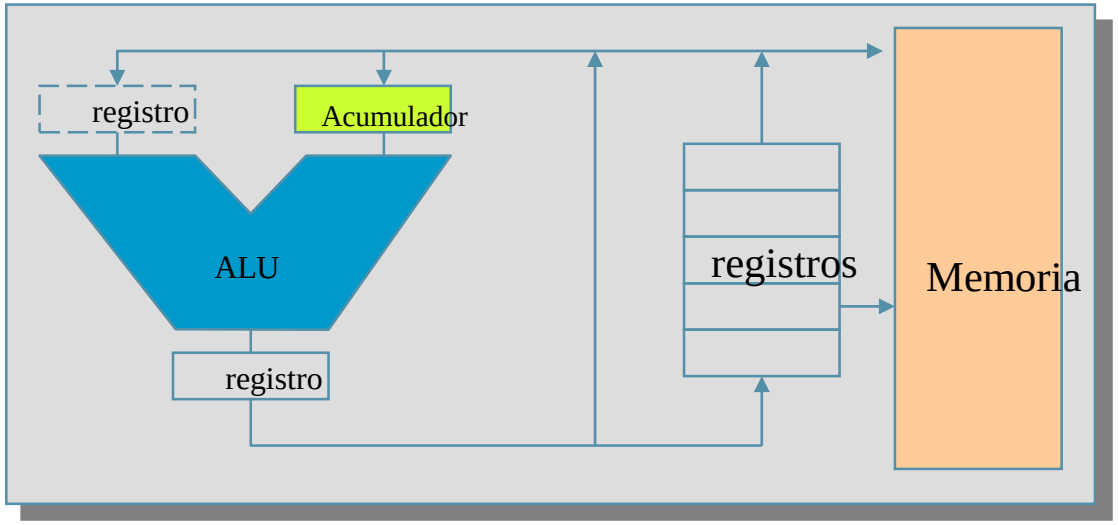
Camino de datos (Datapath) simple de propósito general (4 bits)

Acumulador



Hay que indicar ubicación de un segundo operando. El resultado tambien se almacena en el Acc.

1. Operación
2. Actualizar Acc/FZ

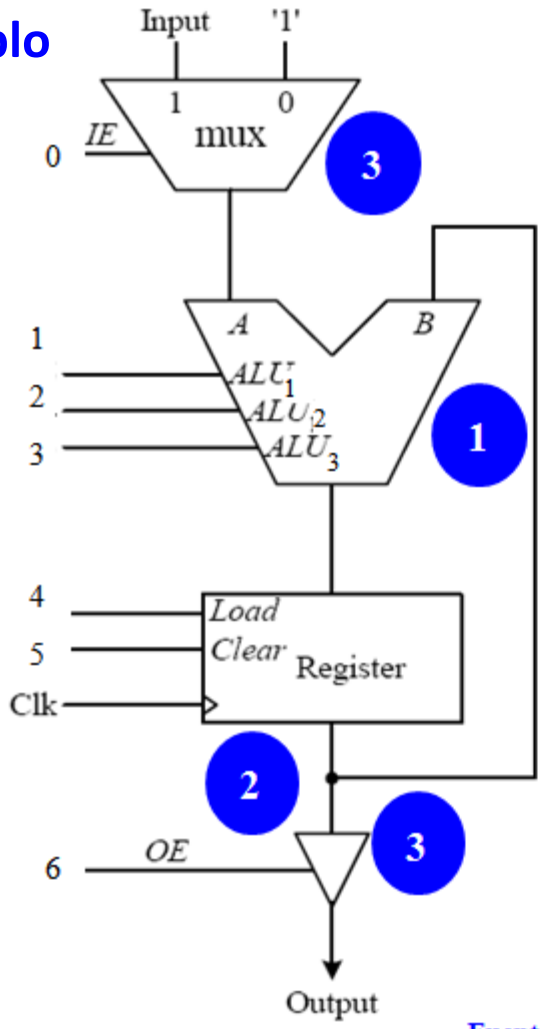


1. Operación
2. Actualiza Reg
3. Actualizar Acc/FZ

Tipos de Camino de datos

Camino de datos (Datapath) simple de propósito general (4 bits)

Ejemplo



ALU_3	ALU_2	ALU_1	Operation
0	0	0	Pass through A
0	0	1	A AND B
0	1	0	A OR B
0	1	1	NOT A
1	0	0	$A + B$
1	0	1	$A - B$
1	1	0	$A + 1$
1	1	1	$A - 1$

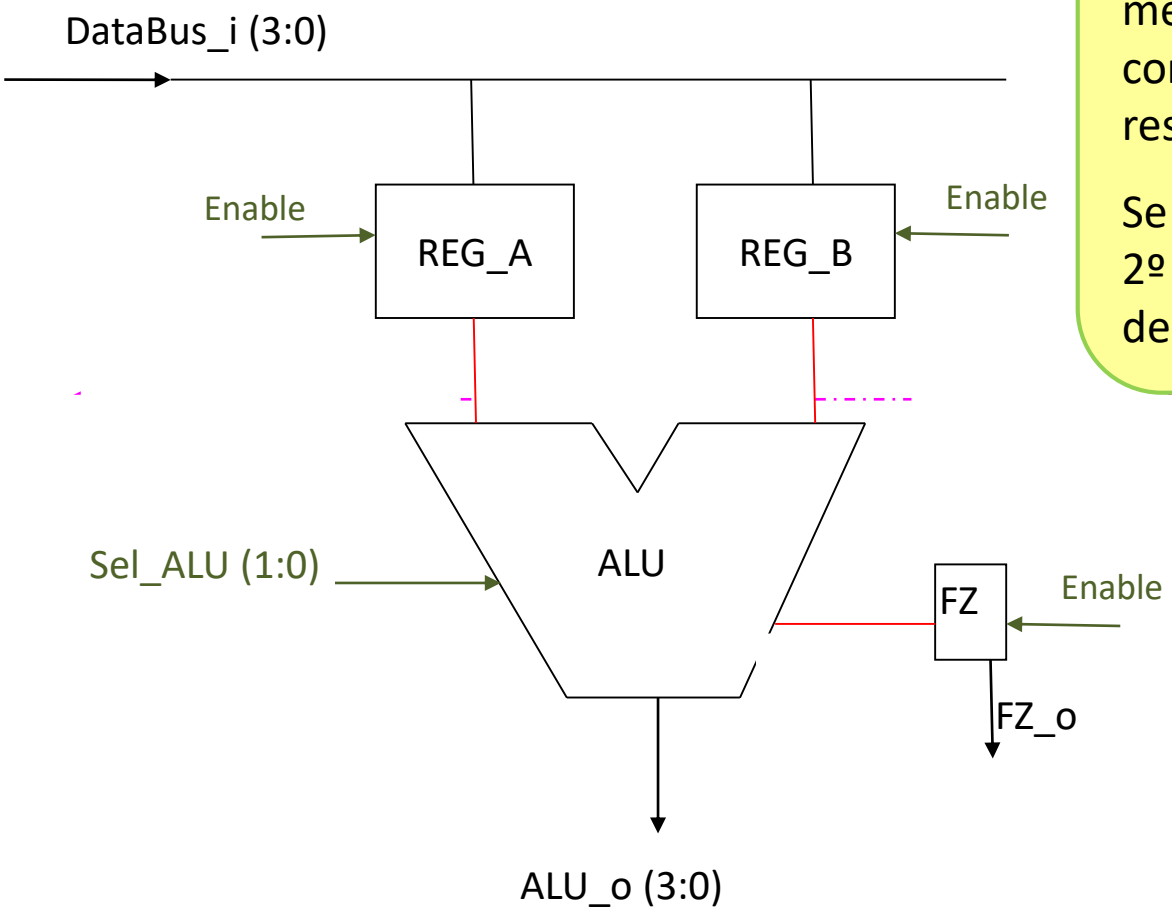
(b)

Fuente: Microprocessor Design: Principles and Practices With VHDL. Enoch O. Hwang

Tipos de Camino de datos

Camino de datos (Datapath) simple de propósito general (4 bits)

Memoria-memoria



Hay que indicar dirección de memoria de A y de B, asi como dónde almacenar el resultado.

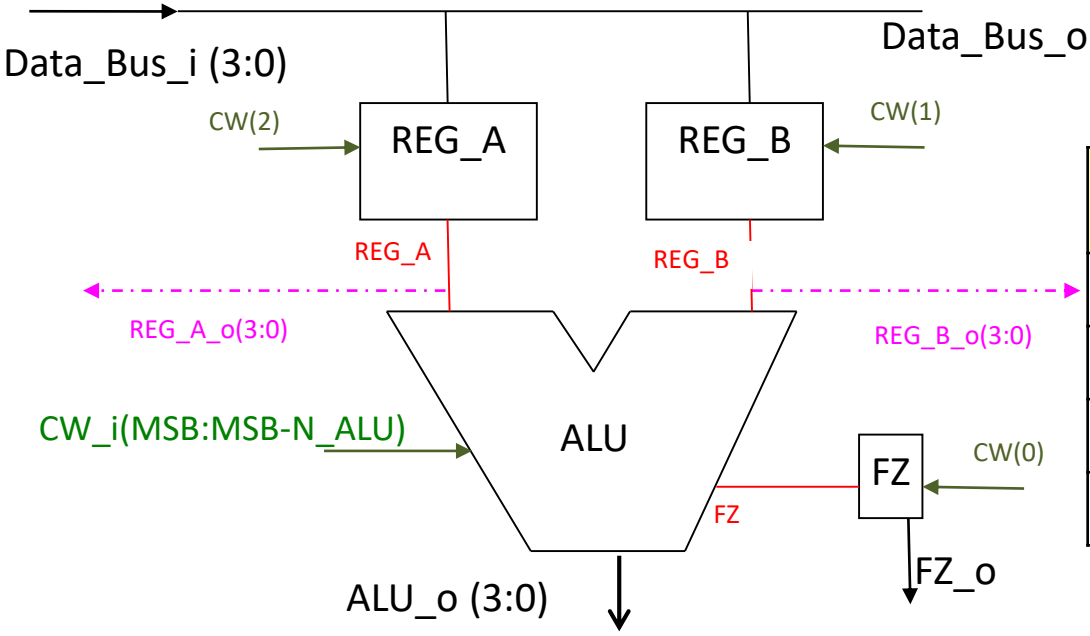
Se puede hacer coincidir el 2º operando con dirección del resultado.

1. Guardar en REGA
2. Guardar en REGB
3. Seleccionar operación /Actualizar FlagZero

4.2.1. Camino de datos de DidaComp (DataPath_0)

UCA
Universidad de Cádiz

Camino de datos de Didacomp



		Funcion ALU
0	0	Paso de A
0	1	Inc A
1	0	A + B
1	1	A - B

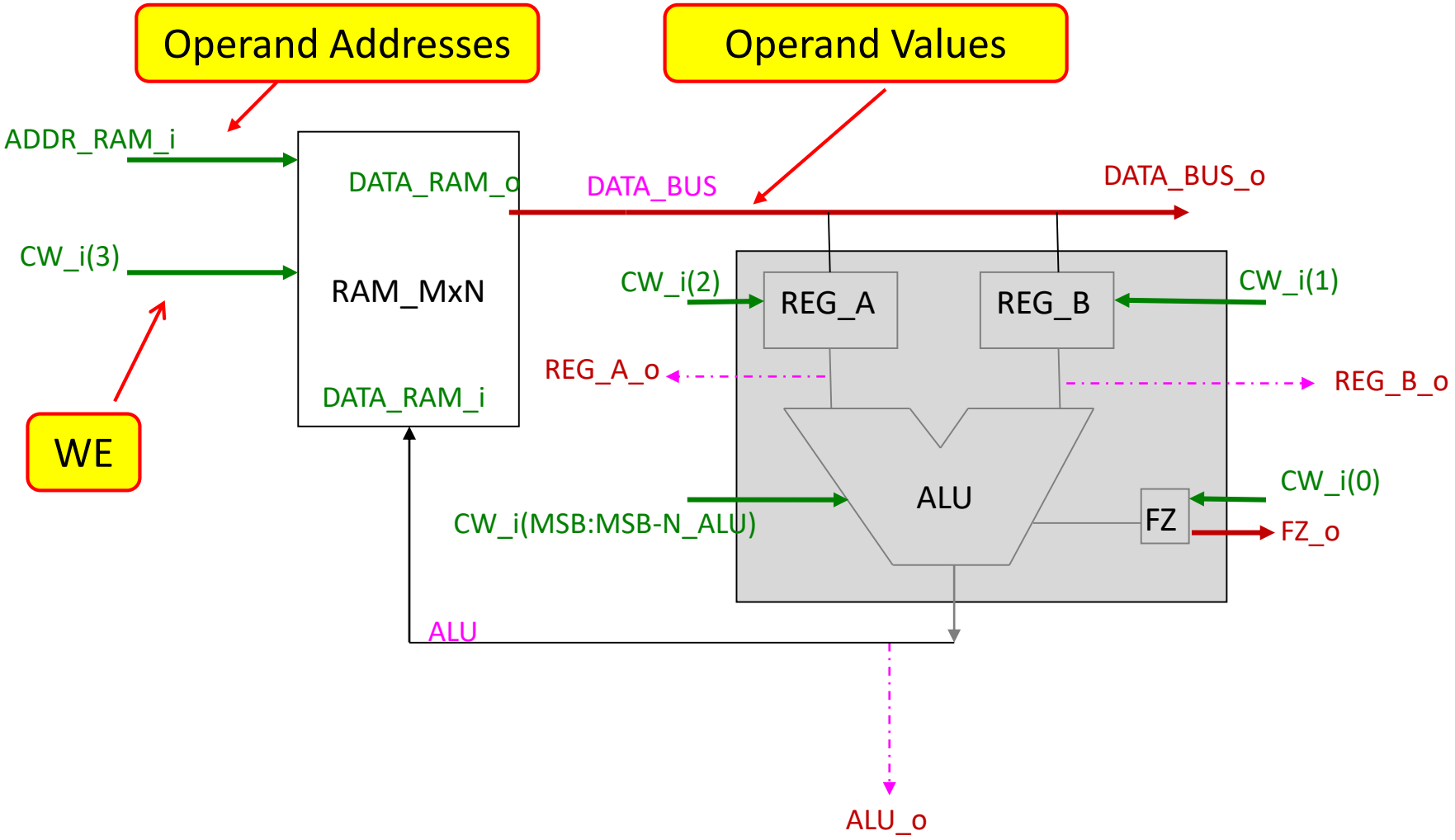
CLK	Bus de control				
	Microinstrucción	**CW(5-4)**	**CW(2)**	**CW(1)**	**CW(0)**
		ALU ope.	Carga REG_A	Carga REG_B	Carga FZ
1	Carga OPE1 desde RAM al registro A	XX	1	0	0
2	Carga OPE2 desde RAM al registro B	XX	0	1	0
3	Seleccionar suma en ALU	10	0	0	1

4.4. Añadiendo RAM a DidaComp

UCA
Universidad de Cádiz

Añadiendo RAM a DidaComp

Origen de los datos, memoria RAM



Añadiendo RAM a DidaComp

Ejecución de instrucciones: Líneas de control (Control Word o CW)

Instrucción "Suma dos datos ADD A,B" → 5 microinstrucciones, una por ciclo de reloj.

CLK	Bus de control					
	Microinstrucción	**CW(5-4)**	**CW(3)**	**CW(2)**	**CW(1)**	**CW(0)**
		ALU OPE	!R/W RAM	Carga REG_A	Carga REG_B	Carga FZ
1	Establecer dirección del operando A	XX	0	0	0	0
2	Carga OPE1 desde RAM al registro A	XX	0	1	0	0
3	Establecer dirección del operando B	XX	0	0	0	0
4	Carga OPE2 desde RAM al registro B	XX	0	0	1	0
5	Seleccionar suma en ALU	10	1	0	0	1

4.4. Diseño del juego de instrucciones de DidaComp

UCA
Universidad de Cádiz

Juego de instrucciones de DidaComp

Formato de Instrucciones Didacomp (Harvard, Memoria-Memoria)

Código de operación (OP_COD)	Dirección MEMO OPEA (addr1)	Dirección MEMO OPEB/RES (addr2)
?	b(ADDR_RAM-1:0)	b(ADDR_RAM-1:0)

Instrucciones

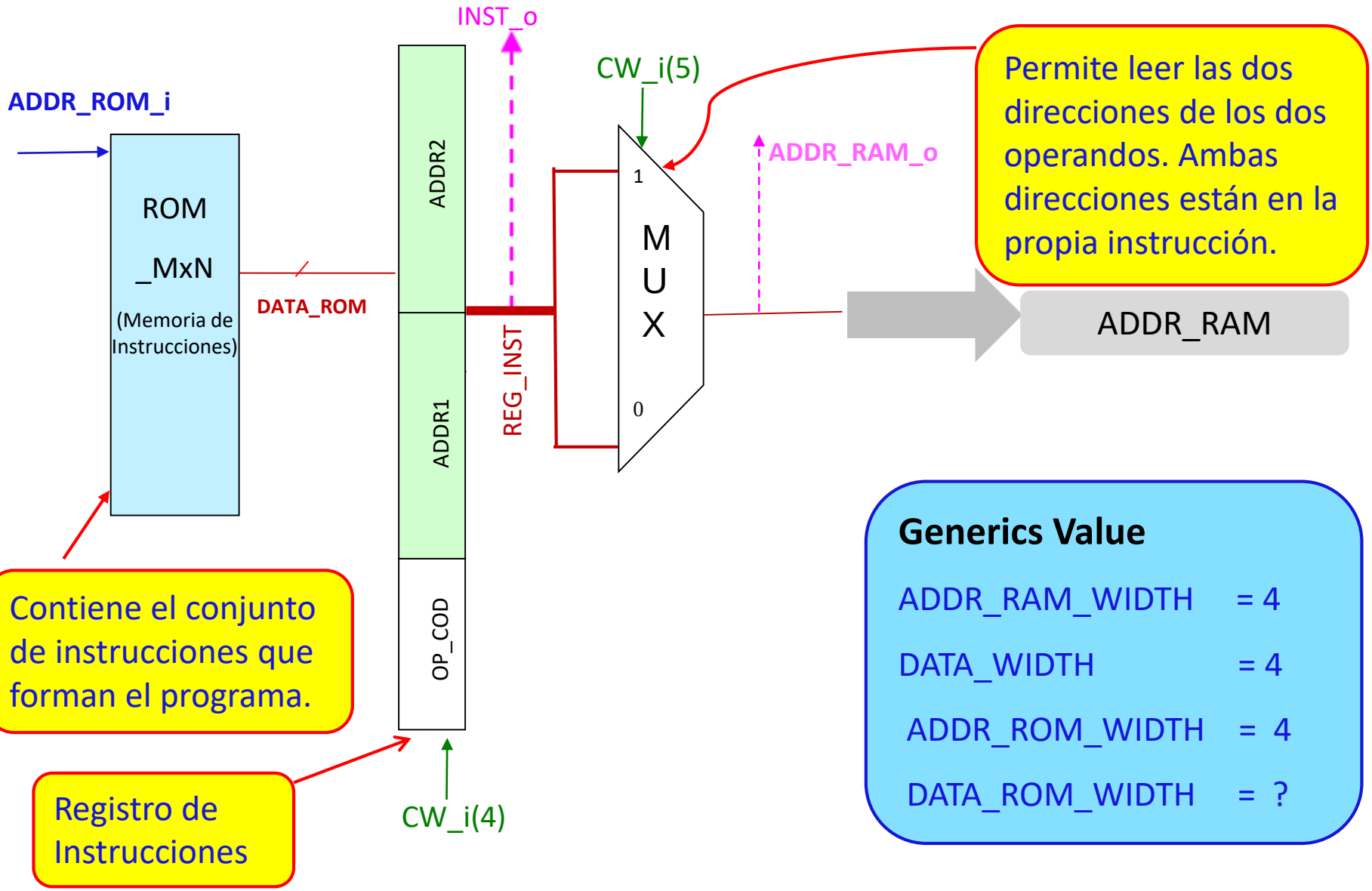
Aunque no es necesario, usaremos un OP_COD de tamaño 4 bits para mantener la potencia de 2.

Instrucción	OP_COD	Función de la instrucción
MOV addr1, addr2	0000	[addr1] → addr2
INC addr1	0001	[addr1] + 1 → addr1
ADD addr1,addr2	0010	[addr1] + [addr2] → addr2
SUB addr1,addr2	0011	[addr1] - [addr2] → addr2
???	0100	???

4.5. Añadiendo la memoria ROM de programa a DidaComp (ROM + RAM + DataPath_0)

UCA
Universidad de Cádiz

Memoria de programa (ROM) y Registro de instrucciones



Generics Value

ADDR_RAM_WIDTH	= 4
DATA_WIDTH	= 4
ADDR_ROM_WIDTH	= 4
DATA_ROM_WIDTH	= ?

Memoria de programa ROM de DidaComp

Instrucción "Suma dos datos ADD A,B"

CLK	Bus de control							
	Microinstrucción	CW(7-6)	CW(5)	CW(4)	CW(3)	CW(2)	CW(1)	CW(0)
		ALU OPE	Sel. Addr. Ope.	Lee y guarda Inst (RI)	!R/W RAM	Carga REG_A	Carga REG_B	Carga FZ
1	(Hay ya una dirección en ROM) Lee instrucción en ROM. La guarda en RI	XX	0	1	0	0	0	0
2	Establece dirección RAM del operando A	XX	0	0	0	0	0	0
3	Carga OPEA desde RAM al registro A	XX	0	0	0	1	0	0
4	Establecer dirección RAM del operando B	XX	1	0	0	0	0	0
5	Carga OPEB desde RAM al registro B	XX	1	0	0	0	1	0
6	Seleccionar suma en ALU, escribe resultado en RAM, actualiza FZ	10	1	0	1	0	0	1

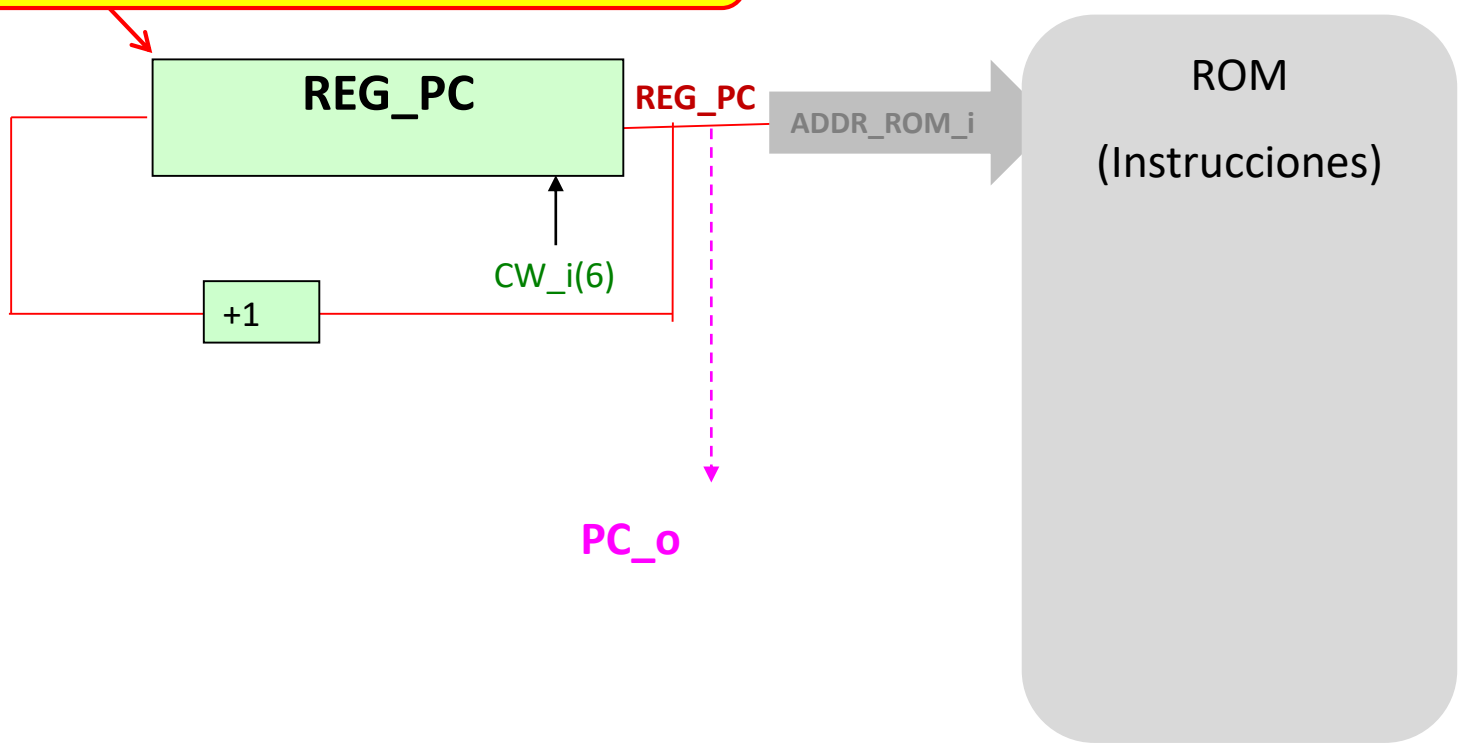
4.6. Contador de programa de DidaComp

UCA
Universidad de Cádiz

Contador de programa de DidaComp

Contador de Programa (PC)

Registro contador de programa: almacena la dirección de la instrucción a leer.



4.7. Estructura de salto de DidaComp

UCA
Universidad de Cádiz

Estructura de salto de DidaComp

Redefinición del Formato de Instrucciones: Estructura de salto (BEZ)

Código de operación	Dirección OPE1 (addr1)	Dirección OPE2/RES (addr2)
b10 b9 b8	b7 b6 b5 b4	b3 b2 b1 b0
Código de operación		**Dirección de salto**
100	X X X X	**b3 b2 b1 b0**

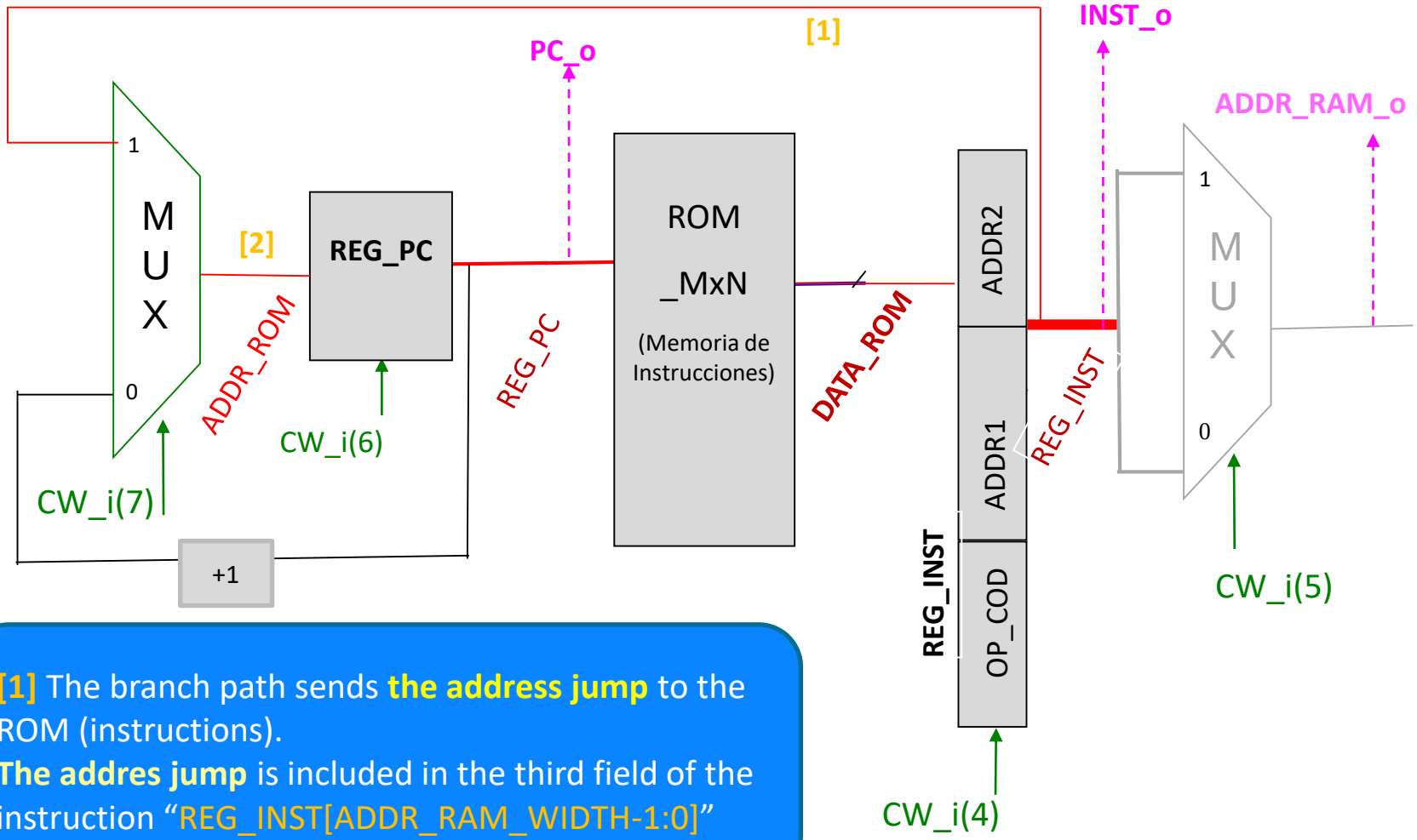
Instrucciones

Aunque no es necesario, usaremos un OP_COD de tamaño 4 bits para mantener la potencia de 2.

Instrucción	OP_COD	Función de la instrucción
MOV addr1, addr2	0000	[addr1] → addr2
INC addr1	0001	[addr1] + 1 → addr1
ADD addr1,addr2	0010	[addr1] + [addr2] → addr2
SUB addr1,addr2	0011	[addr1] - [addr2] → addr2
BEZ addr2	0100	PC ← addr2 (FZ=1)

Estructura de salto de DidaComp

[2] Now, there are two sources of address for the ROM (instructions)



[1] The branch path sends **the address jump** to the ROM (instructions).
The addres jump is included in the third field of the instruction "REG_INST[ADDR_RAM_WIDTH-1:0]"

Memoria de programa ROM de DidaComp

Instrucción "Suma dos datos ADD A,B"

CLK	Bus de control									
	Microinstrucción	CW(9-8)	CW(7)	CW(6)	CW(5)	CW(4)	CW(3)	CW(2)	CW(1)	CW(0)
		ALU OPE	Sel. Fuente Addr. Inst	Incrementa reg PC	Sel. Addr. Ope.	Lee y guarda Inst (RI)	!R/W RAM	Carga REG_A	Carga REG_B	Carga FZ
1	Sel. Addr. ROM. Guarda INST en RI. Incrementa reg PC	XX	0	1	0	1	0	0	0	0
2	DECO/Establece dirección RAM del operando A	XX	0	0	0	0	0	0	0	0
3	Carga OPEA desde RAM al registro A	XX	0	0	0	0	0	1	0	0
4	Establecer dirección RAM del operando B	XX	0	0	1	0	0	0	0	0
5	Carga OPEB desde RAM al registro B	XX	0	0	1	0	0	0	1	0
6	Seleccionar suma en ALU, escribe resultado en RAM, actualiza FZ	10	0	0	1	0	1	0	0	1