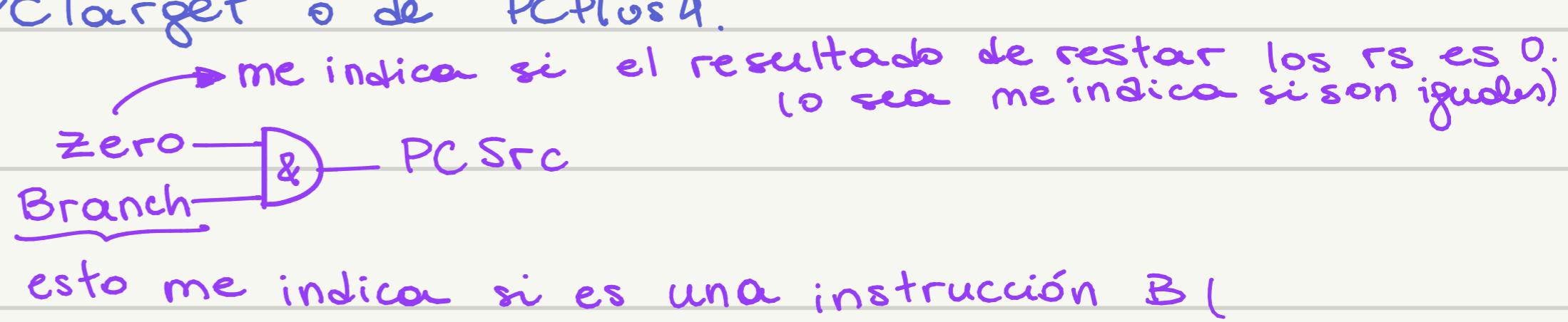


Resumen de Microarquitectura (Final de Sistemas Digitales)

Señales de Control.

PC Src = Indica si el PC proviene de PCTarget o de PCPlus4.

$$PC\ Next = \begin{cases} PC\ Plus\ 4 & \text{si } PC\ Src = 0 \\ PC\ Target & \text{si } PC\ Src = 1 \end{cases}$$



esto me indica si es una instrucción B (

Result Src = me indica qué valor voy a guardar en el registro destino de la instrucción

$$WD3 = \begin{cases} ALU\ Result & \text{si } Result\ Src = 0 \\ Read\ Data(RD) & \text{si } Result\ Src = 1 \end{cases}$$

Para ello también es necesario el RegWrite=1

MemWrite = Me indica si en la dirección dada por el ALUResult voy a escribir el valor del rs2 (RD2)

ALUSrc = Me indica, según la operación, si necesita que el ALU use el imm o el rs2 de la instrucción

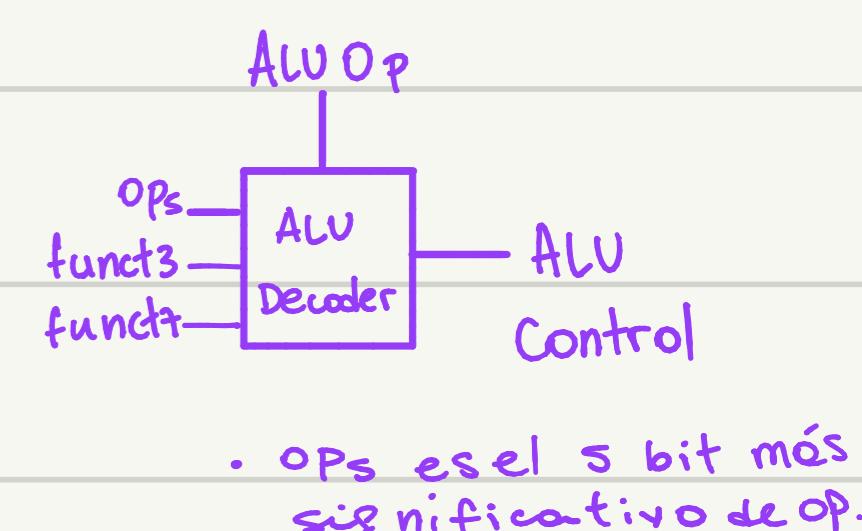
(AluOp + decode(func3, funct7))

ALUControl: determina la operación realizada por el ALU

ALU Control (Posibles valores) ALU OP

000 (add, lw, sw)
001 (sub, beq)
010 (and)
011 (or)
101 (slt)

00 { si vale 00 sé que es porque la instrucción es lw o sw y el ALUControl=000 (add)
01 { si vale esto es porque la instrucción es beq y ALUControl=001 (sub)
10 { si vale esto, para determinar cual instrucción es necesario ver funct3 y op_s, funct7

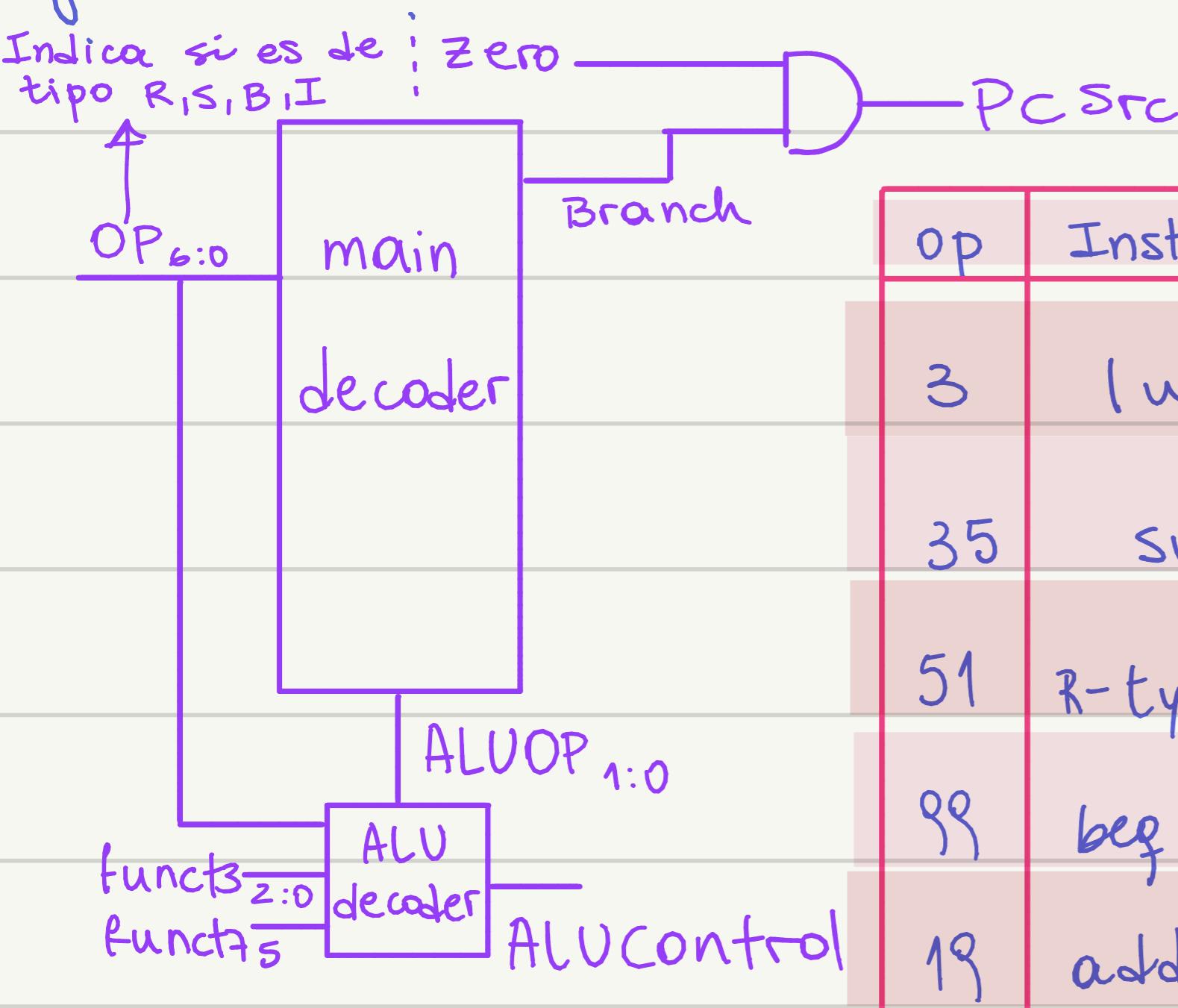


• op_s es el 5 bit más significativo de op.

ImmSrc: indica el tipo de partición del inmediato en la instrucción

Tipo	Valores	Bits
I	00	12
S	01	12
B	10	13

RegWrite: Indica si voy a guardar o no la información de WD3 en el registro destino de la instrucción (A3). si vale 1 escribo en A3, si vale 0, no.



OP	Instr.	RegWrite	ImSrc	ALUSrc	MemWrite	ResultSrc	B	ALUOP
3	lw	1	00	1	0	1	0	00
35	sw	0	01	1	1	x	0	00
51	R-type	1	xx	0	0	0	0	10
99	beq	0	10	0	0	x	1	01
19	addi	1	00	1	0	0	0	10

Señales de dato (entrada y salida)

1. Program Counter

PC Next = es el resultado del Adder que tiene como entrada al PC y a 4, entonces es $PC + 4$

PC = dirección donde se encuentra almacenada la instrucción.

2. Instruction Memory:

A(entrada) = PC

RD(salida) = Valor almacenado en la dirección indicada por A(PC). Resulta ser una instrucción.

3. Register File : Banco de Registros

A1(entrada) (RS1): 1^{er} Registro fuente de la instrucción (5 bits)

A2(entrada) (RS2): 2^{do} Registro fuente de la instrucción (5 bits)

A3(entrada) (RD): Registro destino de la instrucción (5 bits)

WD3(entrada): lo que voy a escribir en el registro destino en caso de que la instr. lo amerite.

RD1(salida): el valor presente en la dirección indicada por A1.

RD2(salida): el valor presente en la dirección indicada por A2.

4. Extend

ImmExt: inmediato de la instrucción con su signo extendido.

5. ALU

SrcA: RD1

SrcB: Puede ser RD2 o el inmediato con el signo extendido, según lo que indique ALUSrc.

ALU Result

6. Data Memory:

A(entrada): Resultado del ALU

WD(entrada): RD2 lo escribo en la dirección indicada por A si MemWrite = 1.

RD(ReadData): a) Escribo en el registro destino el valor presente en la dirección A si RegWrite = 1 y ResultSrc = 1

b) Si ResultSrc = 0 y RegWrite = 1 entonces escribe el ALUResult en el registro destino.

c) Si RegWrite = 0 entonces no escribe nada en el registro destino porque no hay uno como tal, por ej.: sw.

7. PC Target = PC + imm sirve para determinar el próximo PC.

A partir de la señal de control PCsrc, se decide si realizar el salto (PC+imm)

o no (PC+4). Si PCsrc = 1 entonces PC Next = PC Target, si PCsrc = 0 \Rightarrow PC Next = PC + 4.