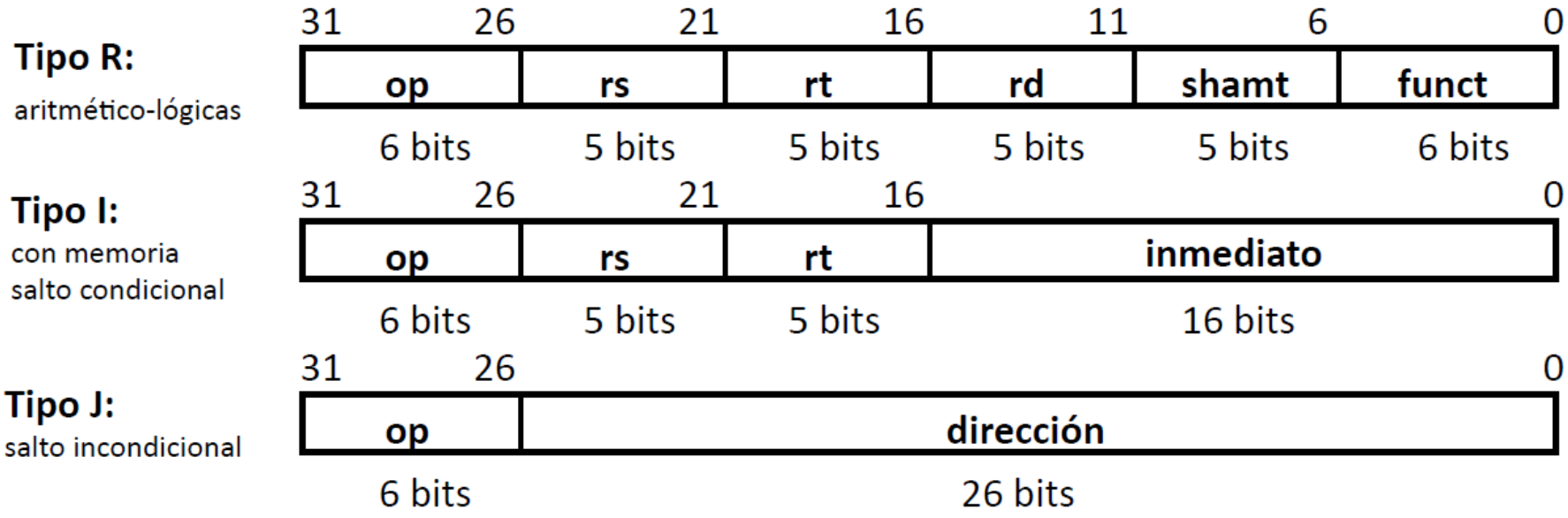
Práctica 6

MIPS multiciclo

Introducción al diseño de un procesador



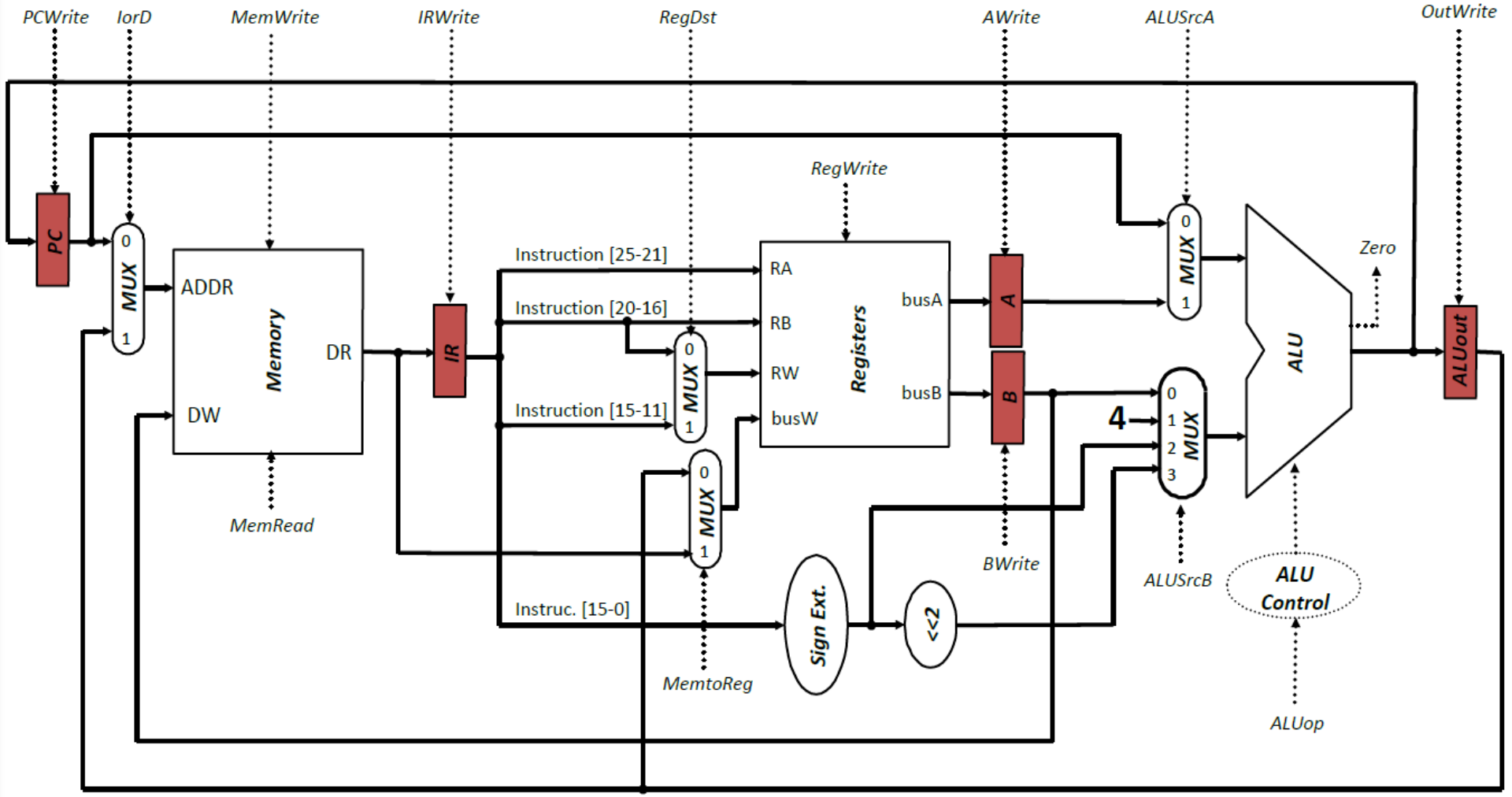
El significado de los campos es:

- **op**: identificador de instrucción
- **rs**, **rt**, **rd**: identificadores de los registros fuentes y destino
- **shamt**: cantidad a desplazar (en operaciones de desplazamiento)
- **funct**: selecciona la operación aritmética a realizar
- **inmediato**: operando inmediato o desplazamiento en direccionamiento a registro-base
- **dirección**: dirección destino del salto

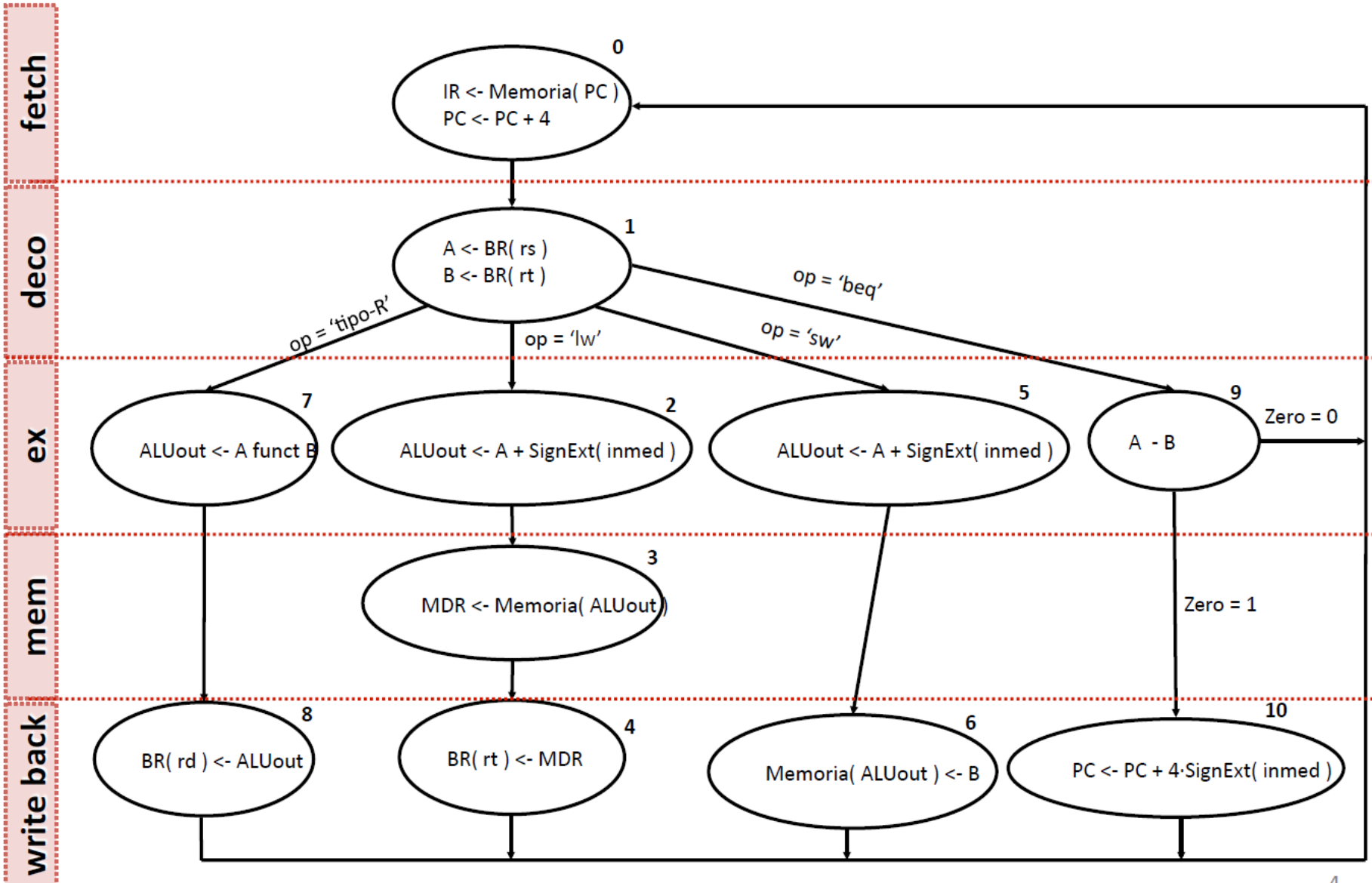
Procesador multiciclo

- **Instrucciones con referencia a memoria**:
  - lw rt, inmed(rs) rt <- Memoria(rs + SignExt(inmed)) , PC <- PC + 4
  - sw rt, inmed(rs) Memoria(rs + SignExt(inmed)) <- rt , PC <- PC + 4
- **Instrucciones aritmético-lógicas con operandos en registros**:
  - add rd, rs, rt rd <- rs + rt, PC <- PC + 4
  - sub rd, rs, rt rd <- rs - rt , PC <- PC + 4
  - and rd, rs, rt rd <- rs and rt , PC <- PC + 4
  - or rd, rs, rt rd <- rs or rt , PC <- PC + 4
  - slt rd, rs, rt (si (rs < rt) entonces (rd <- 1)
    en otro caso (rd <- 0)), PC <- PC+4
- **Instrucciones de salto condicional**:
  - beq rs, rt, inmedsi (rs = rt) entonces (PC <- PC + 4 + 4·SignExp(inmed))
    en otro caso PC <- PC + 4

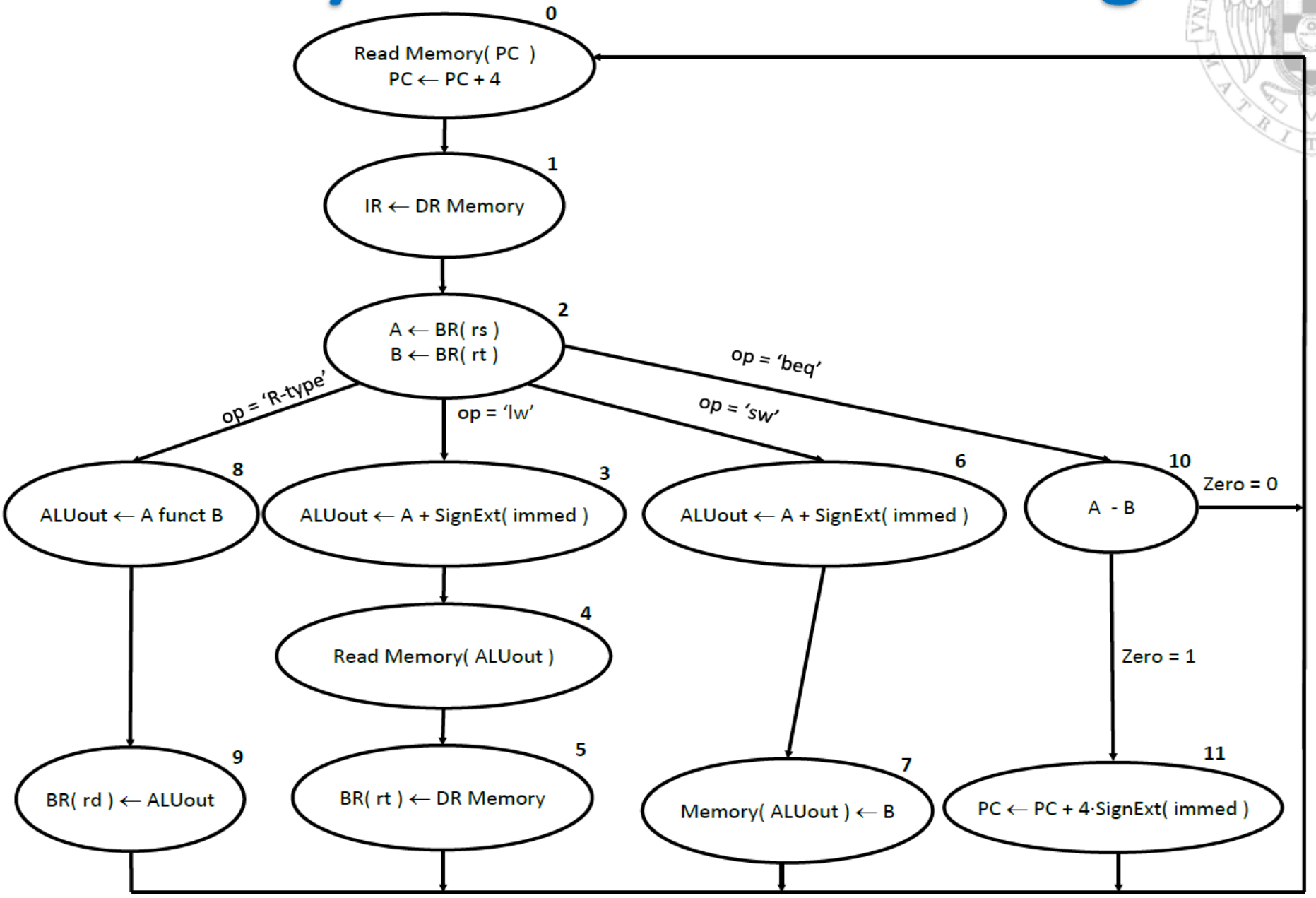
Diseño de la ruta de datos multiciclo



Diseño del controlador multiciclo



Multicycle Control Unit design



Diseño del controlador multiciclo

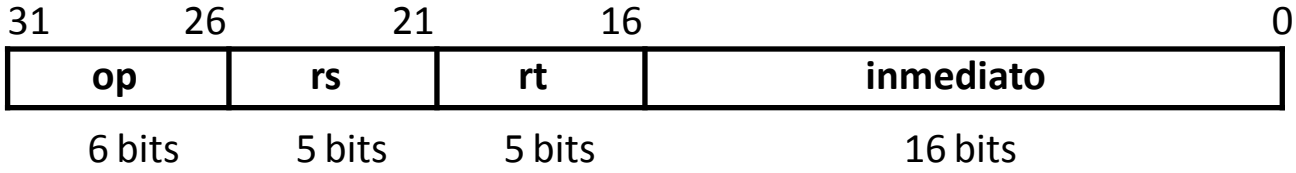
Tabla de verdad del controlador

Estado actual	op	Zero	Estado siguiente	IRWrite	PCWrite	AWrite	BWrite	ALUSrcA	ALUScrB	ALUOp	OutWrite	MemWrite	MemRead	IorD	MDRWrite	MemtoReg	RegDest	RegWrite
0000	XXXXXX	X	0001	1	1			0	01	00 (add)		0	1	0				0
0001	100011 (lw)	X	0010															
0001	101011 (sw)	X	0101	0	0	1	1					0	0					0
0001	000000 (tipo-R)	X	0111															
0001	000100 (beq)	X	1001															
0010	XXXXXX	X	0011	0	0			1	10	00 (add)	1	0	0					0
0011	XXXXXX	X	0100	0	0							0	1	1	1			0
0100	XXXXXX	X	0000	0	0							0	0			1	0	1
0101	XXXXXX	X	0110	0	0		0	1	10	00 (add)	1	0	0					0
0110	XXXXXX	X	0000	0	0							1	0	1				0
0111	XXXXXX	X	1000	0	0			1	00	10 (funct)	1	0	0					0
1000	XXXXXX	X	0000	0	0							0	0			0	1	1
1001	XXXXXX	0	0000	0	0			1	00	01 (sub)		0	0					0
1001	XXXXXX	1	1010															
1010	XXXXXX	X	0000	0	1			0	11	00 (add)		0	0					0

Instrucción move

- **Añadir instrucción move con inmediato:**
  - mv rt, #inmed rt <- SignExt(inmed), PC <- PC + 4

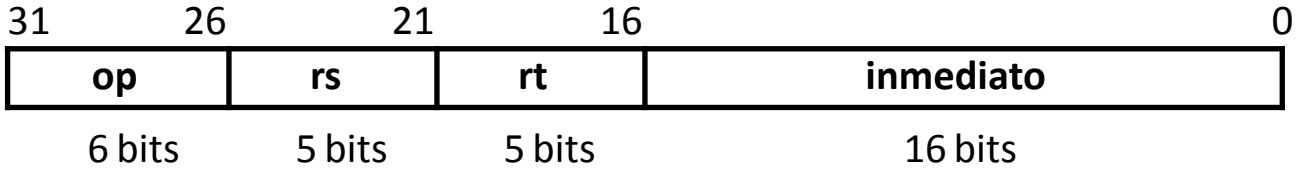
Tipo I:
con memoria
salto condicional



  - Código de operación: “010000”

- **Añadir instrucción move con registro:**
  - mv rt, rs rt <- rs, PC <- PC + 4

Tipo I:
con memoria
salto condicional

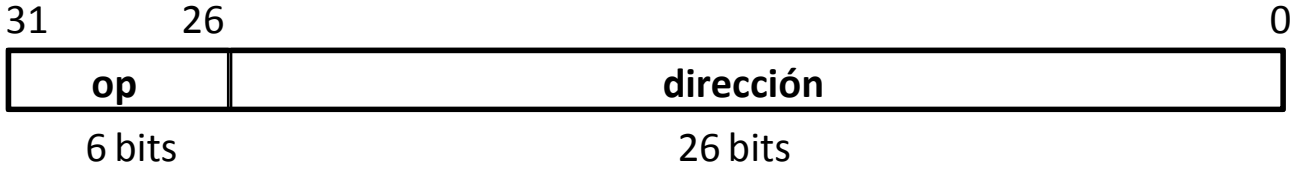


  - Código de operación: “010010”

Instrucción jump (salto incondicional)

- **Añadir instrucción jump:**
  - j dir PC <- “00000000”&dir

Tipo J:
salto incondicional



  - Código de operación: “000010”
- **Modificar el programa dado utilizando la instrucción de salto incondicional**
- **Probar en FPGA que el resultado mostrado por los displays es correcto para los valores introducidos**

Mostrar R3 por displays 7 segmentos

- **Mostrar por los displays 7 segmentos de la placa extendida el contenido del registro R3**
  - Este registro es el que utilizaremos en nuestros programas para almacenar el resultado
  - Se mostrarán los 4 bits menos significativos en el display de menor peso y los 4 bits siguientes en el otro display

Aclaración: No se trata de crear una nueva instrucción, simplemente hay que mostrar el contenido del registro R3 por los displays de 7 segmentos.

- **Probar en FPGA que el resultado mostrado por los displays es correcto**

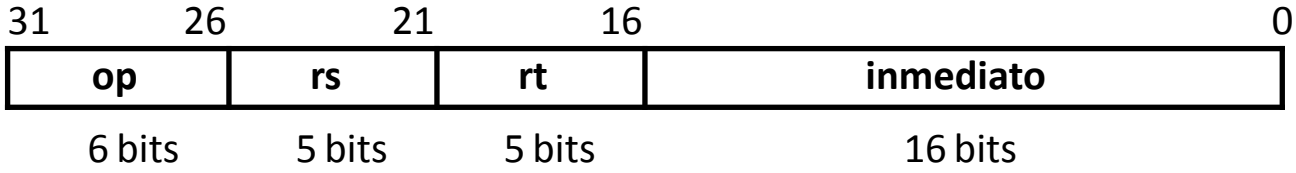
Modo depuración

- **Añadir la funcionalidad de depuración instrucción a instrucción:**
  - Mostrar en el display 7 segmentos de la placa superior el resultado de ((PC – 4) módulo 4) ó (PC[31:2] – 1)
  - Mediante uno de los SW de la placa extendida se seleccionará el modo de funcionamiento:
    - Normal: el programa se ejecuta como hasta ahora instrucción a instrucción sin esperas
    - Depuración: el programa se detiene al comienzo de cada instrucción (estado S1) y permanece así hasta que se presione un pulsador
- **Probar en FPGA que el resultado mostrado por los displays es correcto para los valores introducidos**

Instrucción lectura switches

- **Añadir instrucción lectura switches (SW):**
  - lsw rt, #inmed si (inmed = 0) rt <- SignExt(SW placa extendida[3:0]), PC <- PC + 4
    en otro caso rt <- SignExt (SW placa superior[3:0]), PC <- PC + 4

Tipo I:
con memoria
salto condicional



  - Código de operación: “010001”
- **Modificar el programa dado para que un operando se lea del SW superior y el otro del SW de la placa extendida**
- **Probar en FPGA que el resultado mostrado por los displays es correcto para los valores introducidos**