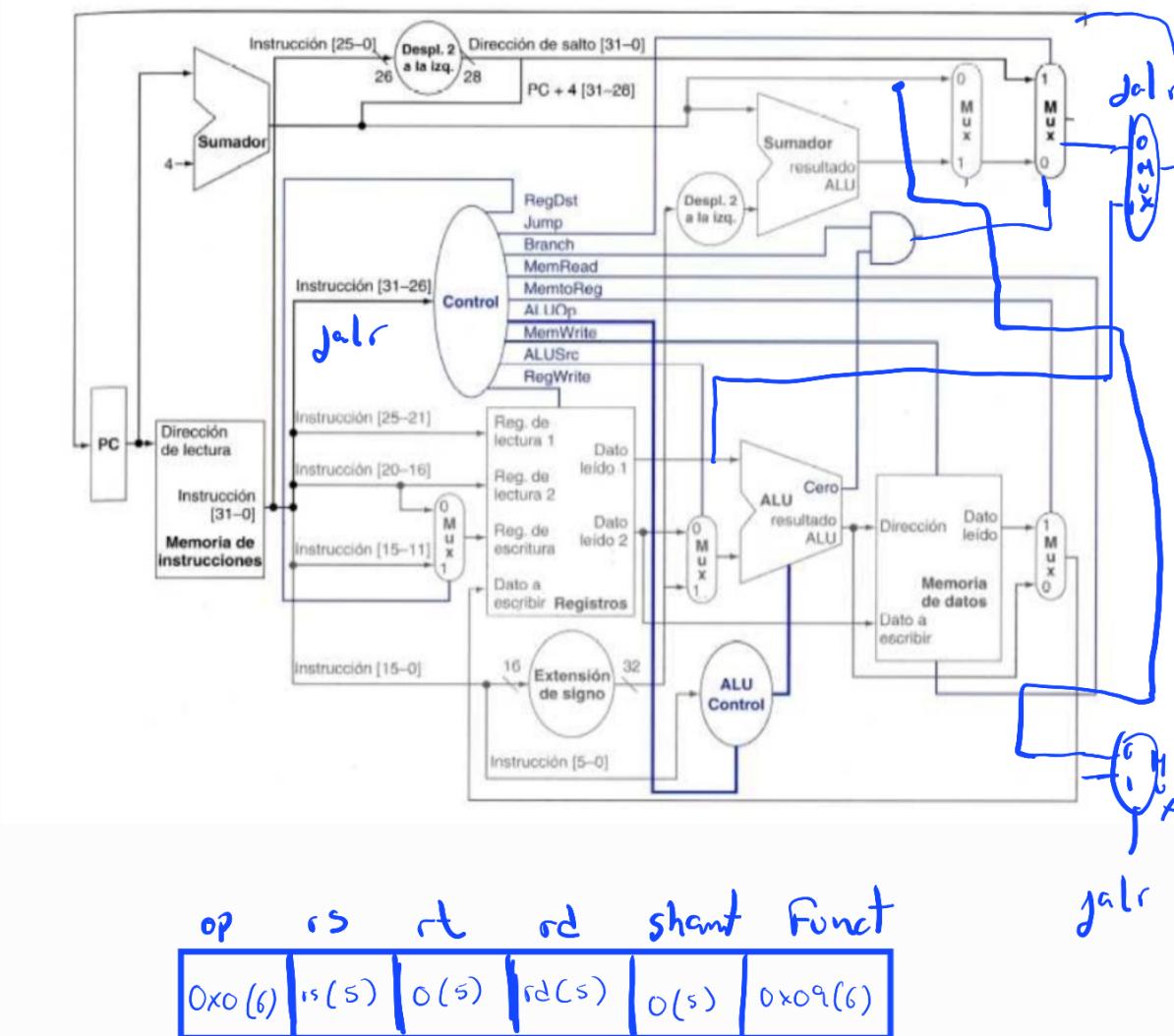


1)

- a) Camino de datos y señales de control necesarias para incorporar la instrucción *jalr* (jump and link mediante registros, rd = PC y PC = rs, es decir la dirección de retorno PC se guarda en el registro rd, y la dirección contenida en el registro rs se carga en el registro PC) para el camino de datos monociclo realizando las modificaciones en la siguiente figura:

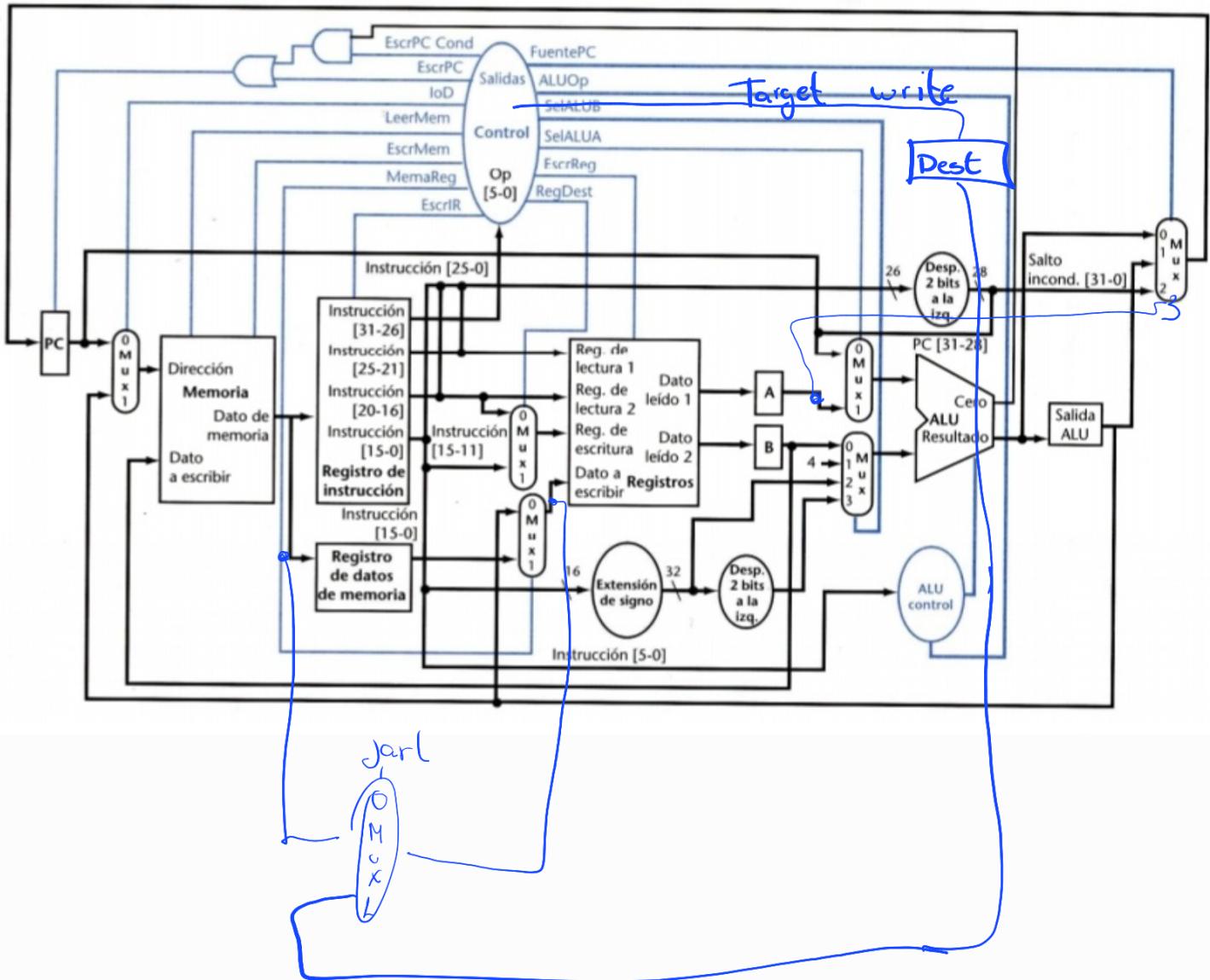


- b) Ampliar la tabla de líneas de control para ver los valores que deben presentar todas las líneas de control que se añadieron en el anterior ejercicio para la instrucción *jalr*.

Instrucción	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	<i>jalr</i>
Formato R	1	0	0	1	0	0	0	1	0	X
lw	0	1	1	1	1	0	0	0	0	X
sw	X	1	X	0	0	1	0	0	0	X
beq	X	0	X	0	0	0	1	0	1	X
<i>jalr</i>	1	X	X	1	0	0	0	X	X	1

2)

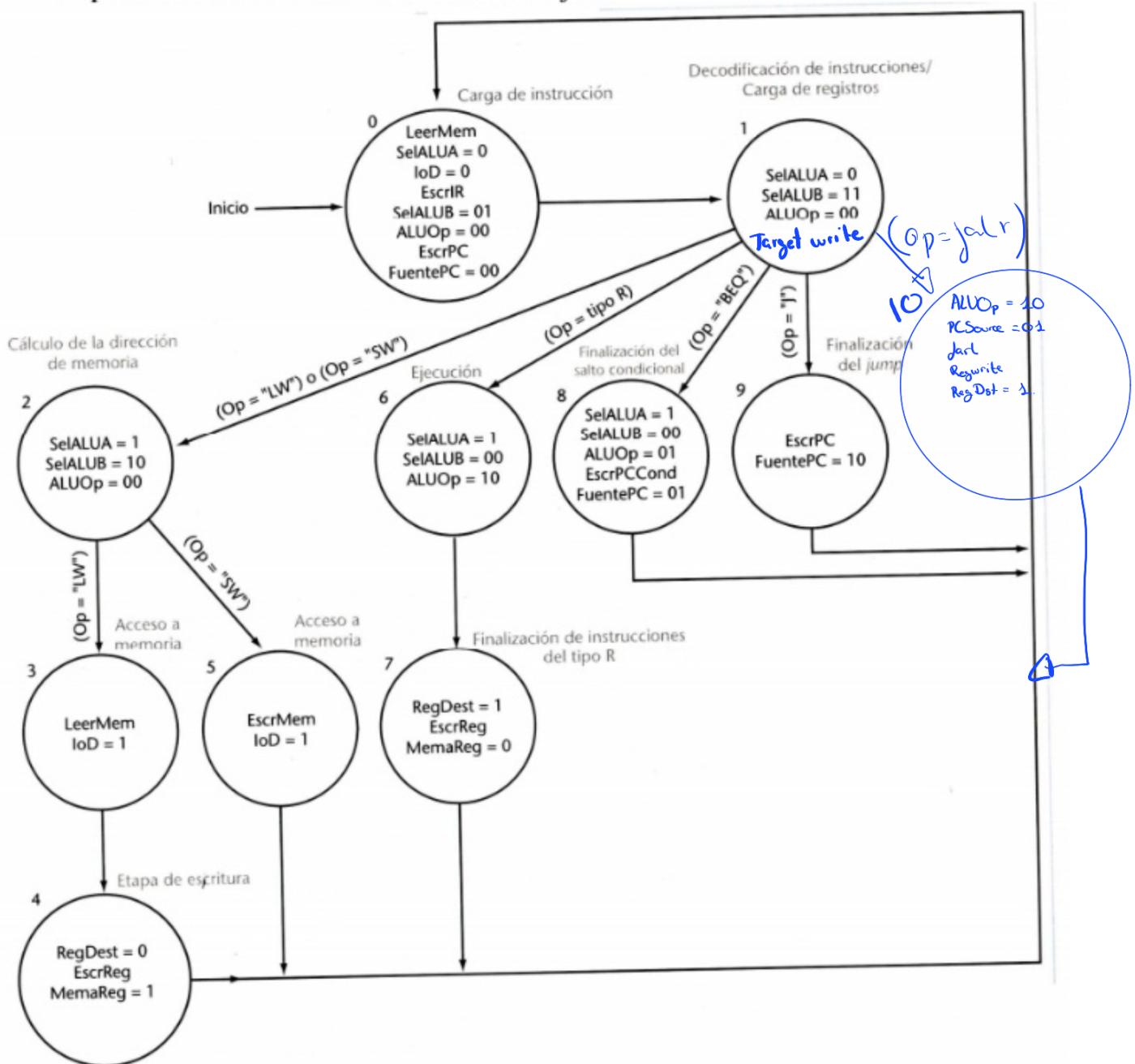
- a) Camino de datos y señales de control necesarias para incorporar la instrucción *jalr* (jump and link mediante registros, rd = PC y PC = rs, es decir la dirección de retorno PC se guarda en el registro rd, y la dirección contenida en el registro rs se carga en el registro PC) para el camino de datos multiciclo realizando las modificaciones en la siguiente figura:



- b) Ampliar la tabla de líneas de control para ver los valores que deben presentar todas las líneas de control que se añadieron en el anterior ejercicio para la instrucción *jarl*.

Nombre de la etapa	Acción para instrucciones del tipo R	Acción para instrucciones de acceso a memoria	Acción para saltos condicionales	Acción para instrucciones Jump
Carga de instrucción		IR = Memoria[PC] PC = PC + 4		<i>jarl</i>
Decodificación de instrucciones/carga de los registros		A = Reg [IR[25-21]] B = Reg [IR[20-16]] SalidaALU = PC + (extensión-signo (IR[15-0]) << 2)		
Ejecución, cálculo de direcciones y finalización de saltos condicionales/jump	SalidaALU = A op B	SalidaALU = A + extensión-signo (IR[15-0])	si (A == B) entonces PC = SalidaALU	PC = PC [31-28] (IR[25-0]<<2)
Acceso a memoria y finalización de instrucciones de tipo R	Reg [IR[15-11]] = SalidaALU	Load: MDR = Memoria[SalidaALU] o Store: Memoria[SalidaALU] = B		Registro [2R[15-11]] = PC + 4 PC = Registro [SR[25-21]]
Finalización de la lectura de memoria		Load: Reg[IR[20-16]] = MDR		

- c) Mostrar lo que se debe añadir a la máquina de estados finitos de la figura para implementar el control de la instrucción *jarl*



3) Las siguientes secuencias de instrucciones hacen uso del camino de datos segmentado del procesador MIPS:

lw	\$1, 40(\$2)
add	\$2, \$3, \$3
add	\$1, \$1, \$2
sw	\$1, 20(\$2)

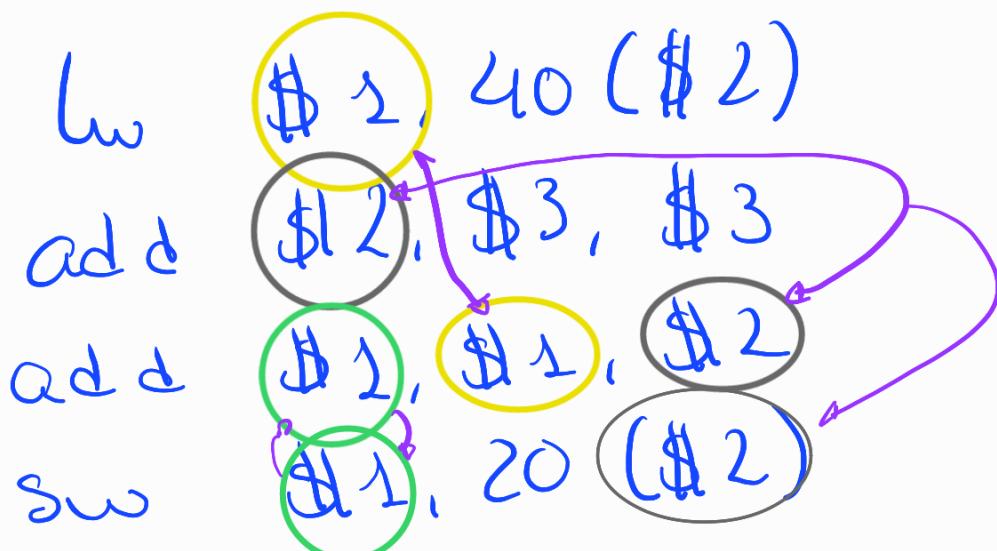
Secuencia 1

add	\$1, \$2, \$3
sw	\$2, 0(\$1)
lw	\$1, 4(\$2)
add	\$2, \$2, \$1

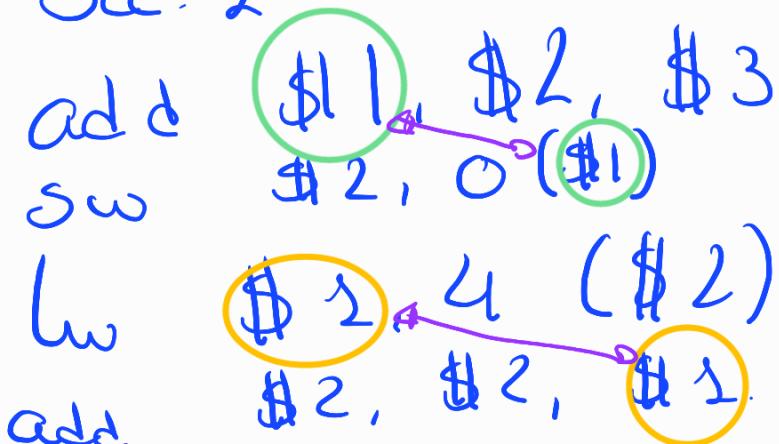
Secuencia 2

- Detectar los posibles riesgos que se producen en las dos secuencias.
- Considerando que no hay anticipación ni detección de riesgos, insertar el menor número de *nop* para asegurar una ejecución correcta para cada una de las secuencias.

a) Sec. 1



Sec. 2



b) Sec 1

No Sim

lw \$2, 40(\$2)

add \$12, \$3, \$3

NOP

NOP

NOP

add \$2, \$1, \$2

NOP

NOP

NOP

sw \$1, 20(\$2)

Sim

lw \$2, 40(\$2)

add \$12, \$3, \$3

NOP

NOP

add \$1, \$1, \$2

NOP

NOP

sw \$1, 20(\$2)

Sec 2

No Sim

addi \$1, \$2, \$3

NOP

NOP

NOP

sw \$2, 0(\$1)

lw \$11, 4(\$2)

NOP

NOP

NOP

add \$2, \$2, \$1

Sim.

addi \$1, \$2, \$3

NOP

NOP

sw \$2, 0(\$1)

lw \$11, 4(\$2)

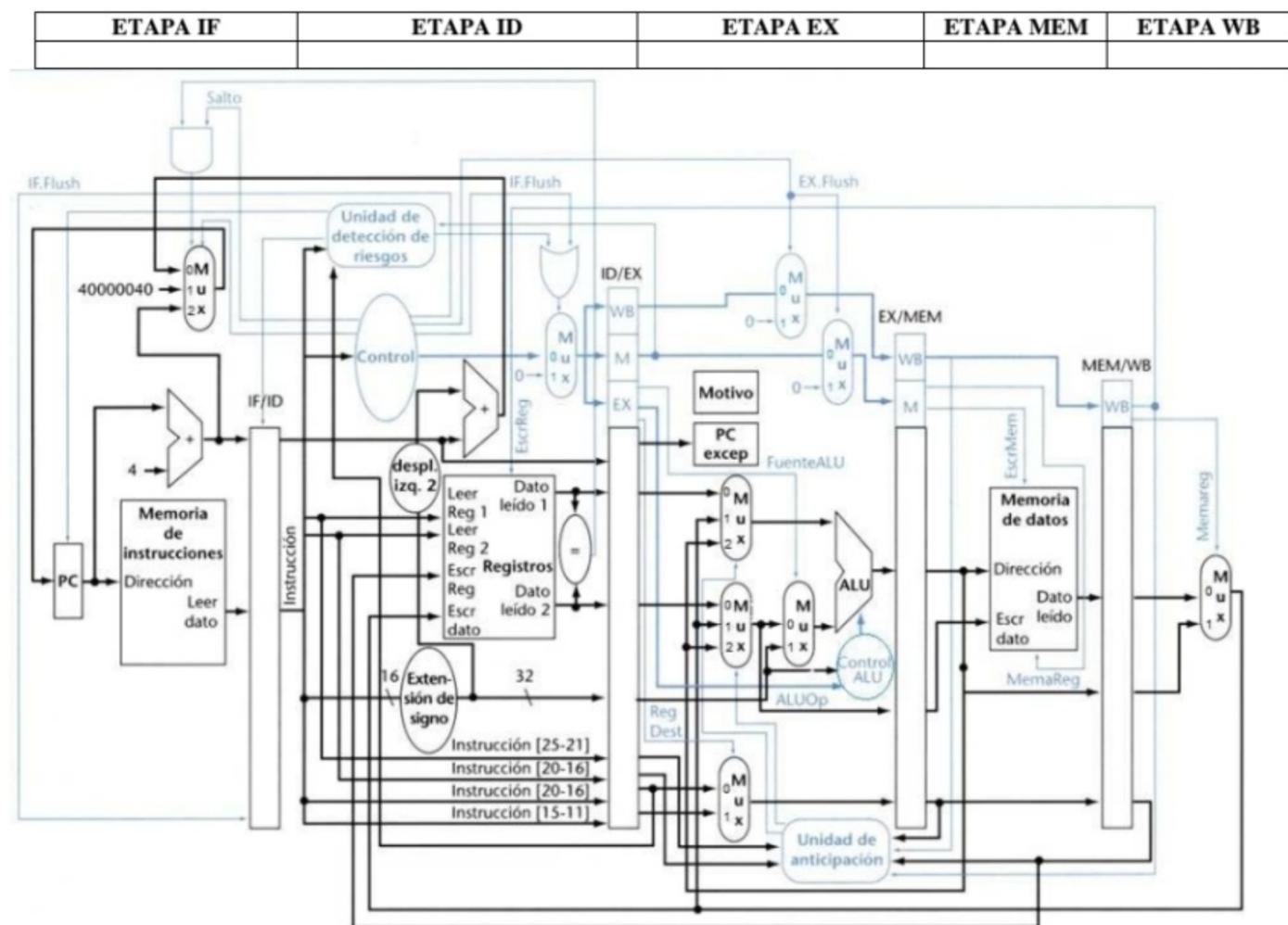
NOP

NOP

add \$2, \$2, \$1

4) Determinar en que situación quedan las etapas de segmentación en el ciclo 5 para las tres secuencias de código 1 del ejercicio anterior, y según los siguientes casos que se proponen:

a) Con unidad de anticipación y detección, con postescritura y lectura simultánea.



Sec. 1

ETAPA IF	ETAPA ID	ETAPA EX	ETAPA MEM	ETAPA WB
—	sw	ADD	ADD	2w

Sec. 2.

ETAPA IF	ETAPA ID	ETAPA EX	ETAPA MEM	ETAPA WB
ADD	NOP	lw	sw	ADD