ARQUITECTURA DE COMPUTADORES

CICLO DE INSTRUCCIÓN

Ing Marlon Moreno Rincón

CICLO DE INSTRUCCIÓN.

Proceso en el que la tarea de ejecución una instrucción se divide en tareas mas pequeñas preestablecidas que debe realizar la CPU para ejecutar una instrucción.

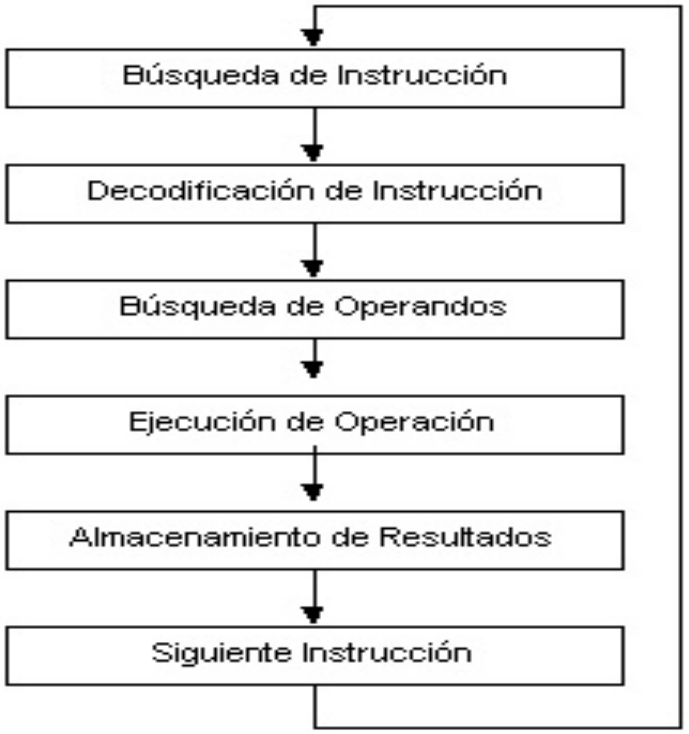
Todas las instrucciones no requieren el mismo numero acciones o tareas para su ejecución.

Un ciclo de instrucción requiere de uno o mas ciclo maquina.

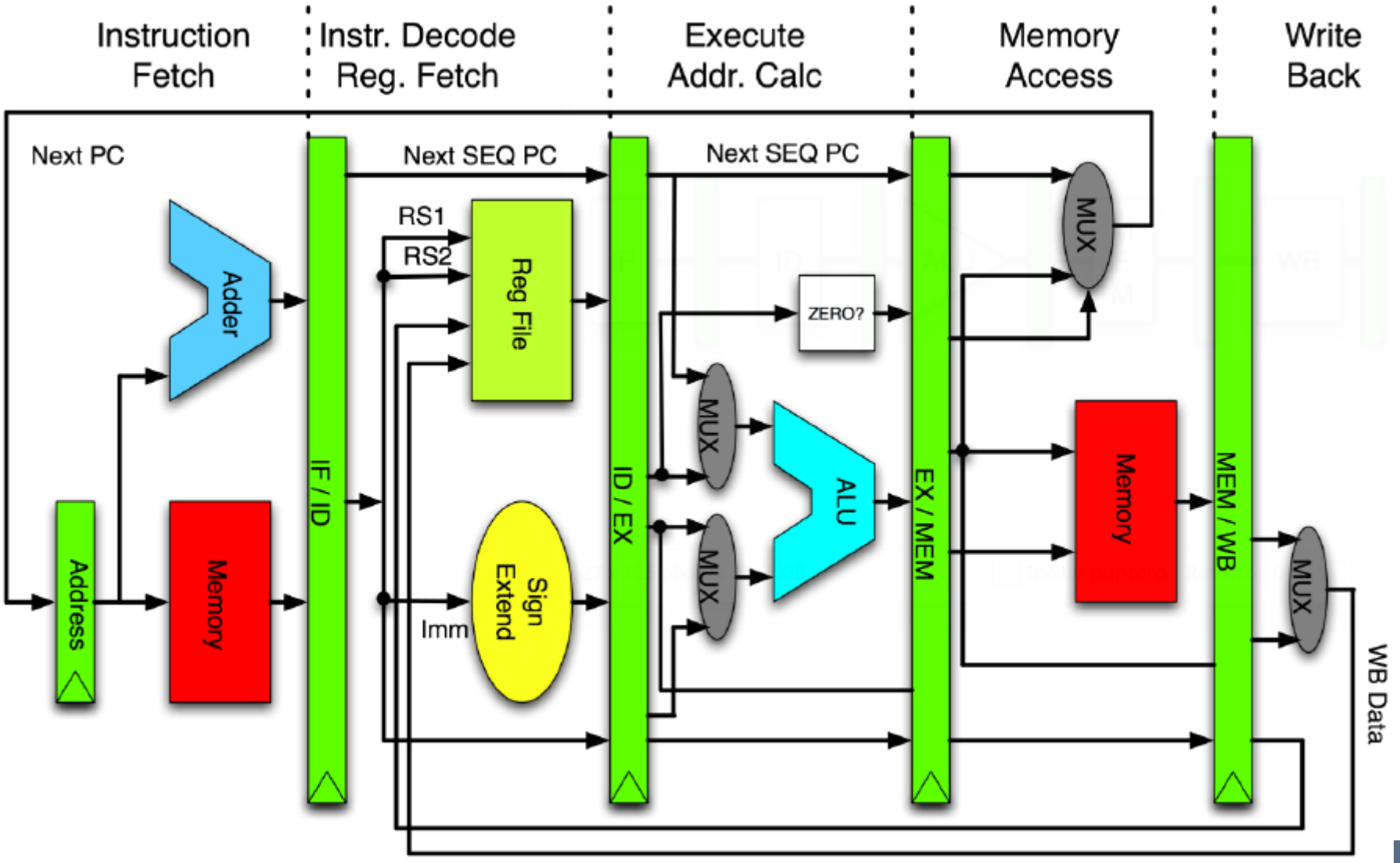
Ciclo maquina: Periodo mínimo que tarda la unidad central de procesamiento en ejecutar una instrucción. Se toma la instrucción que requiere menos tiempo.

CICLO DE INSTRUCCIÓN.

- Instruction fetch (IF)
- Instruction decode (IF)
- Execution (EX)
- Memory read/write (MEM)
- Result write back (WB)



CICLO DE INSTRUCCIÓN.



CICLO DE INSTRUCCIÓN – INSTRUCTION FETCH.

Buscar la instrucción en la memoria principal: Se direcciona la memoria de programa con el dato del contador de programa. La instrucción se guardar en el registro de instrucción actual *(CIR).*

CICLO DE INSTRUCCIÓN – INSTRUCTION DECODE.

Decodificar la instrucción: Se mueve la instrucción de CIR al registro IR. Este registro mantiene la instrucción mientras se ejecuta y permite que se diseccione otra interrupción. La unidad de control decodifica la operación a realizar y mueve los datos de memoria la unidad de procesamiento.

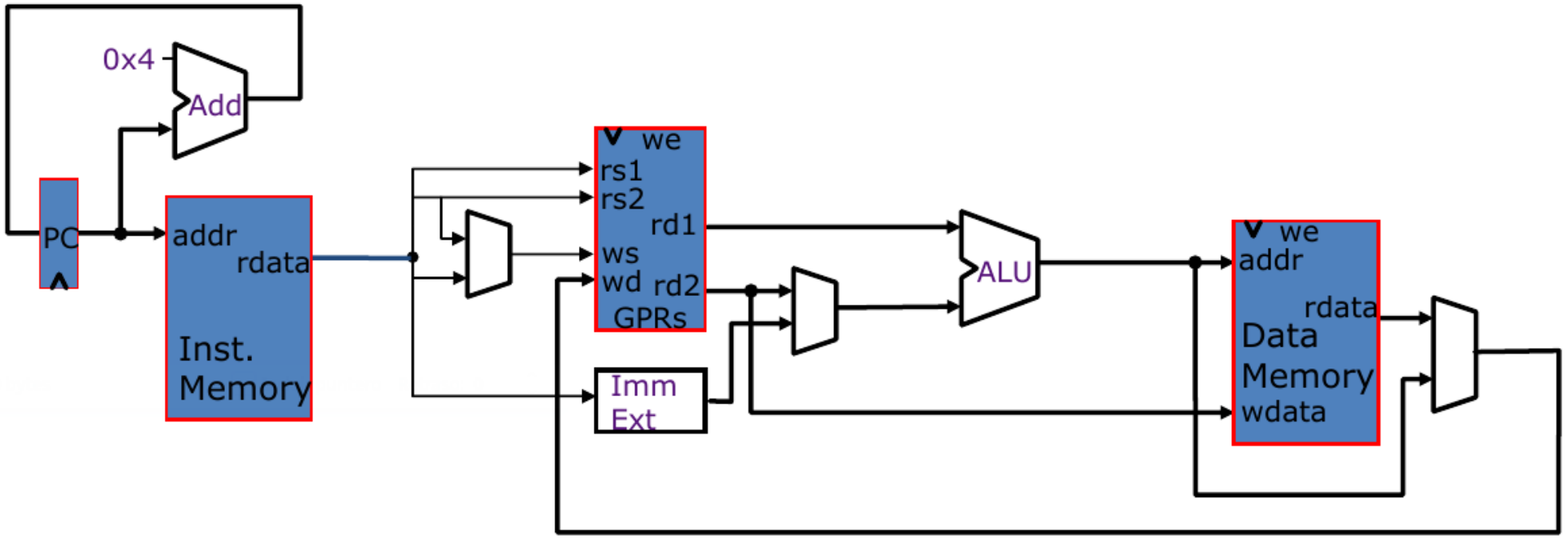
CICLO DE INSTRUCCIÓN – EXECUTION

Ejecución de la instrucción: La unidad de control enviá las señales de control a las unidades de procesamiento para que estas desarrollen la operación.

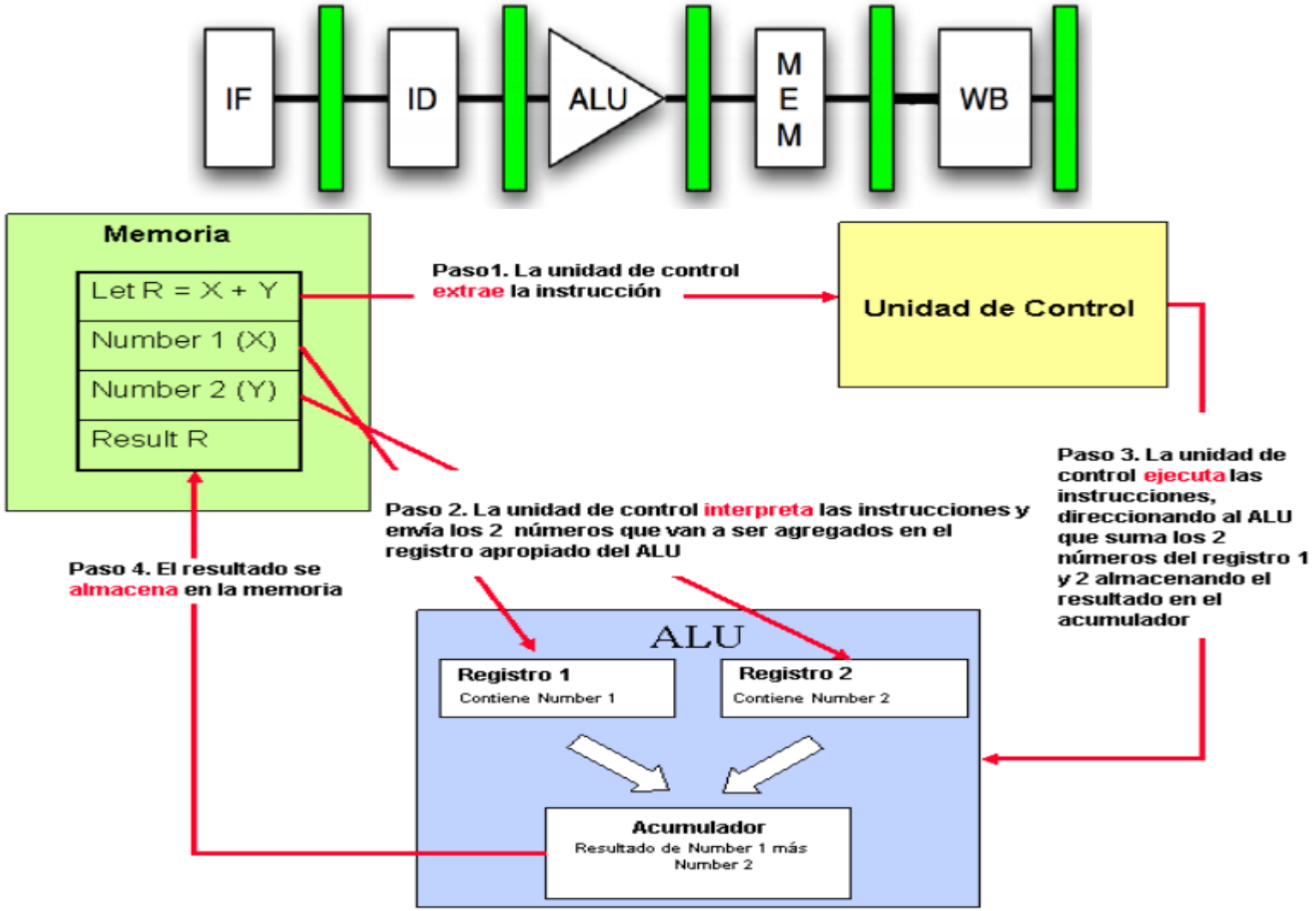
CICLO DE INSTRUCCIÓN – (MEM)-(WB)

Almacenar o guardar resultados: El o los resusltados son alamacenados en memoria o enviados a los diferentes dispositivos. El contador de programa se incrementa o se modifica deacuerdo a los resultados obtenidos de la operación.

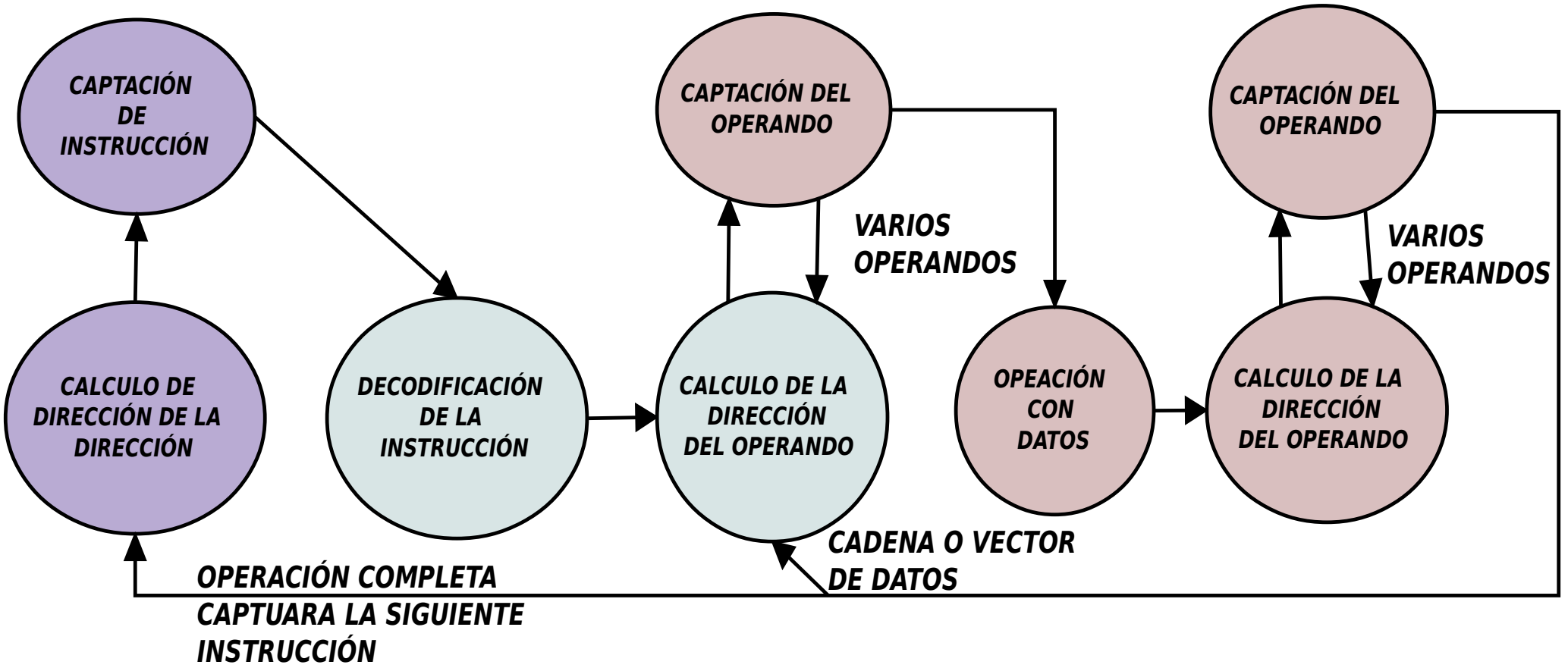
ARQUITECTURA BASICA



CICLO DE INSTRUCCIÓN.



CICLO DE INSTRUCCIÓN

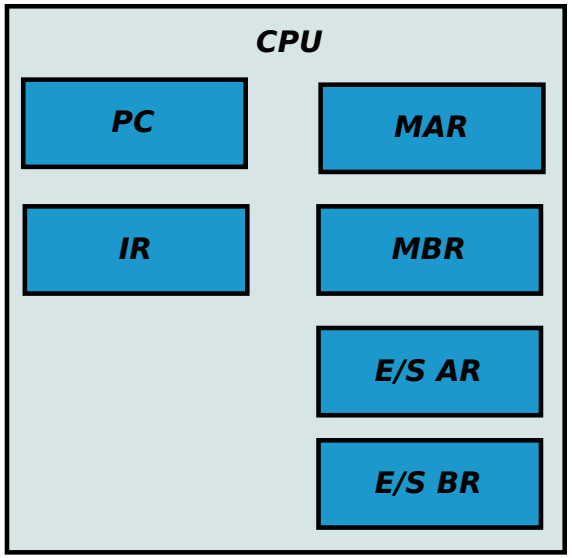


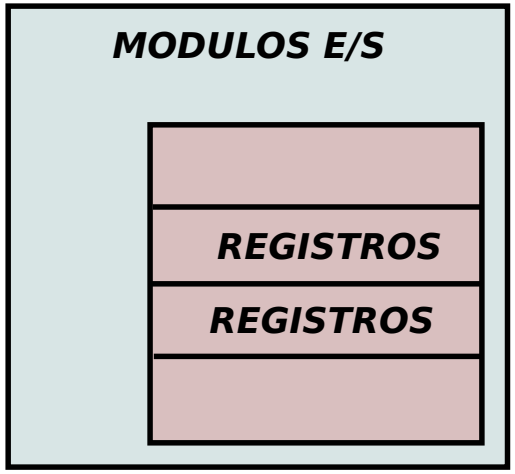
EJEMPLO

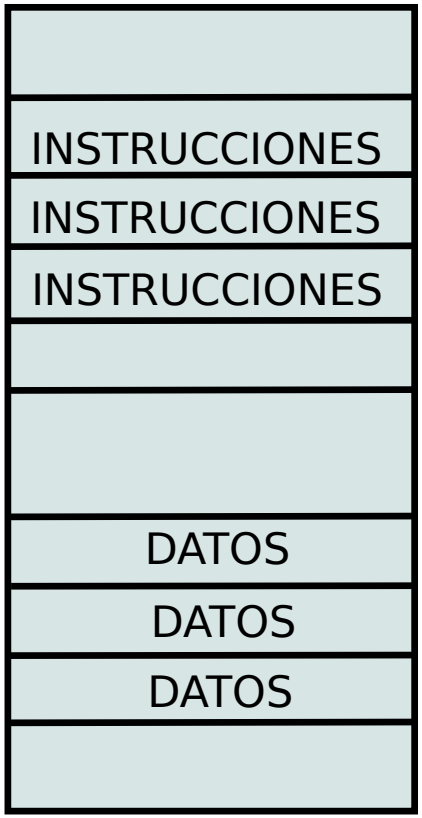
Considere una maquina hipotética, con un set de instrucciones de longitud fija, de 16 bits para instrucciones y datos. Utiliza un único registro para el almacenamiento y operación de datos llamando acumulador (AC).

Utilizando la información de la imagen de la diapositiva 13 realizar paso a paso la ejecución del programa de suma entre dos números, resaltando los ciclos de instrucción que se presentan en cada instrucción usada en el programa.

EJEMPLO.







PC = CONTADOR DE PROGRAMA
IR = REGISTRO DE INSTRUCCIONES
MAR = REGISTRO DE DIRECCIÓN DE MEMORIA
MBR = REGISTRO DE BUFFER DE MEMORIA
E/S AR =REGISTRO DE DIRECCIONES DE E/S
E/S BR =REGISTRO BUFFER DE E/S

EJEMPLO

Formato de instrucción

Formato de datos

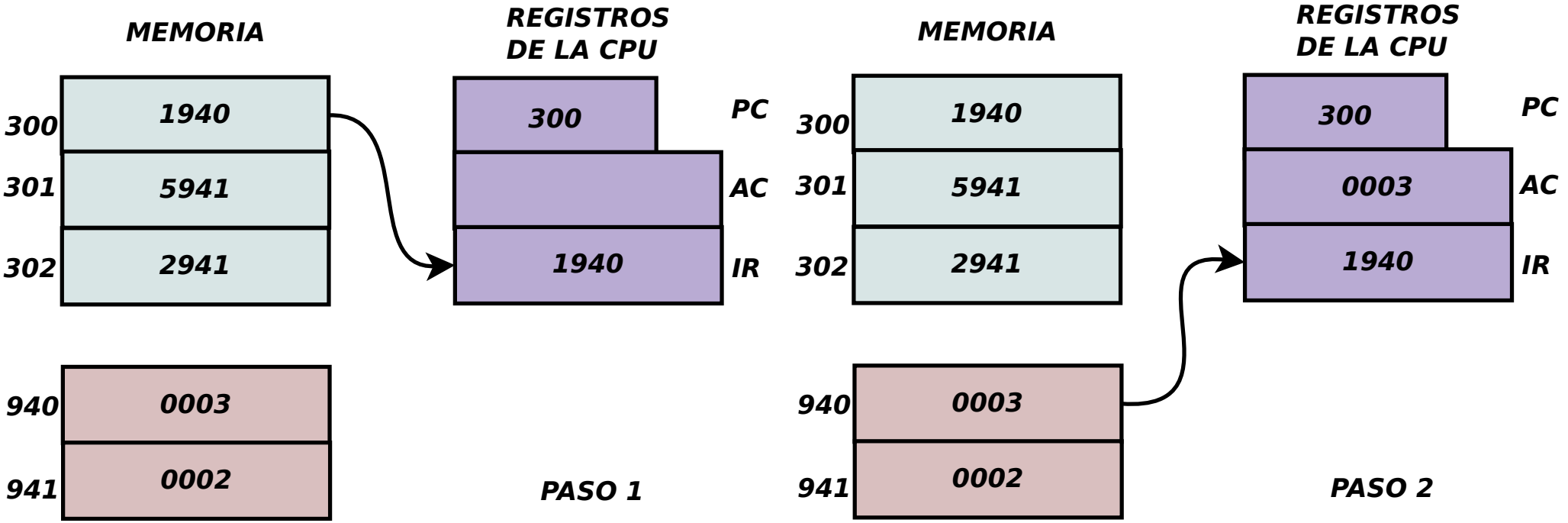
Lista parcial de codop

0001 = Cargar AC desde memoria

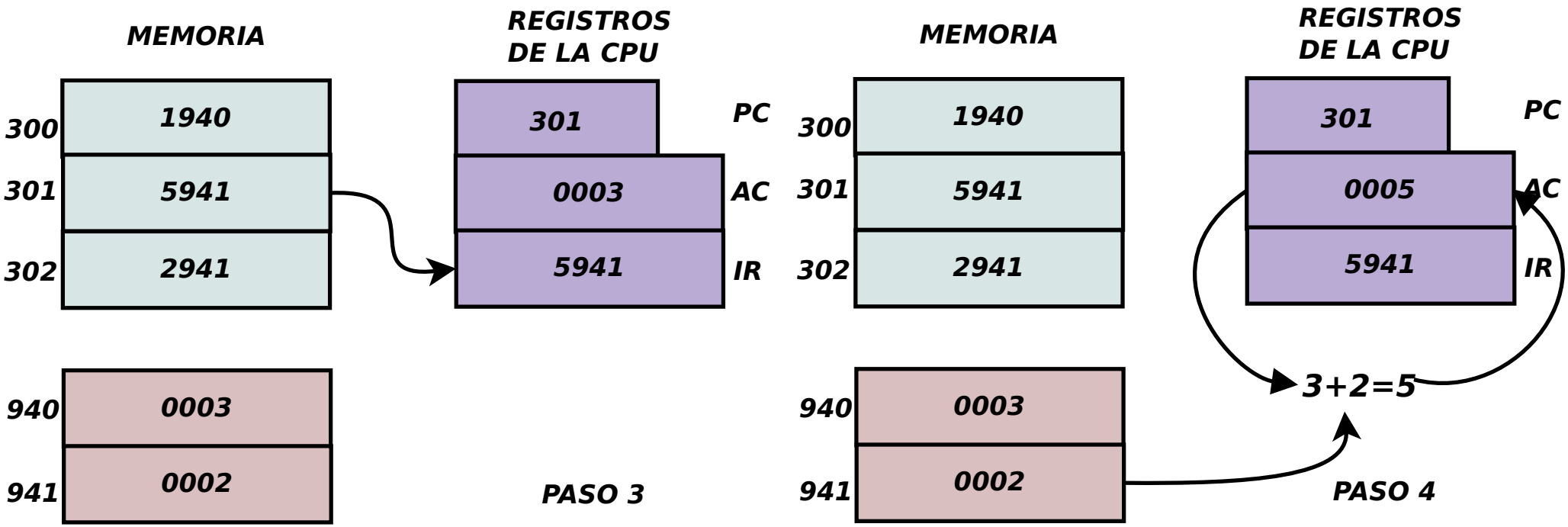
0010 = Almacenar AC en memoria

0101 = Sumar a AC un dato de memoria

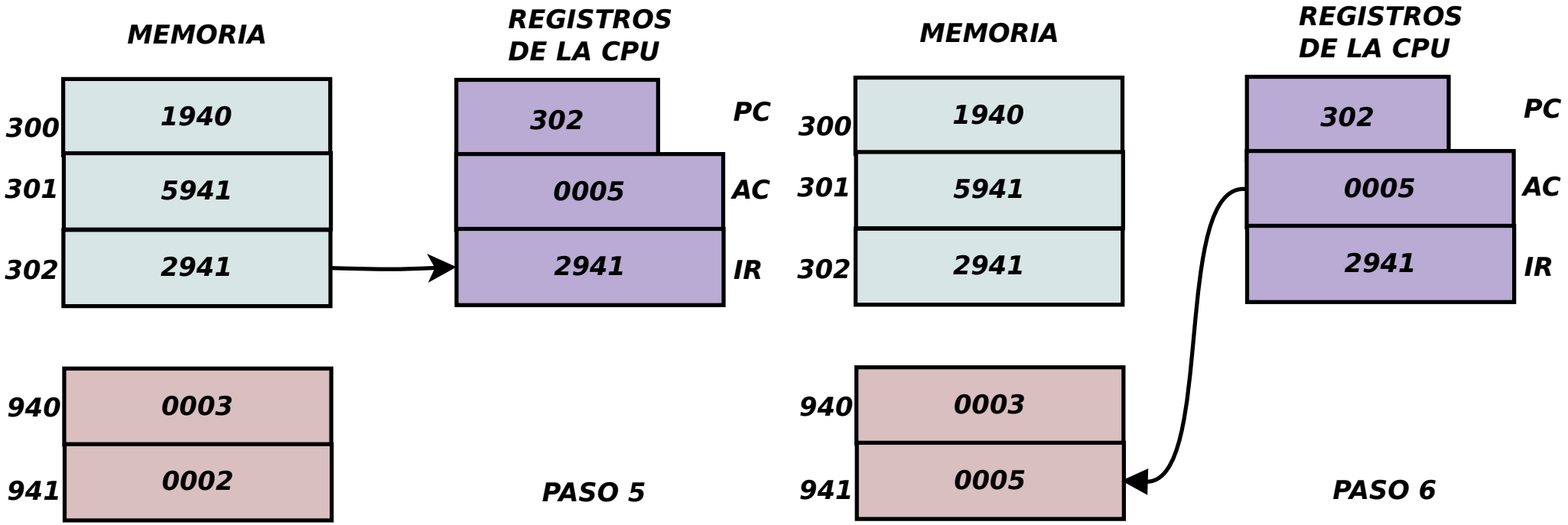
EJEMPLO


MEMORIA
REGISTROS DE LA CPU
300
1940
301
5941
302
2941
940
0003
941
0002
PC
AC
IR
PASO 1
MEMORIA
REGISTROS DE LA CPU
300
1940
301
5941
302
2941
940
0003
941
0002
PC
0003
AC
IR
PASO 2

EJEMPLO


MEMORIA
REGISTROS DE LA CPU
300
1940
301
5941
302
2941
940
0003
941
0002
301
PC
0003
AC
5941
IR
PASO 3
MEMORIA
REGISTROS DE LA CPU
300
1940
301
5941
302
2941
940
0003
941
0002
301
PC
0005
AC
5941
IR
3+2=5
PASO 4

EJEMPLO

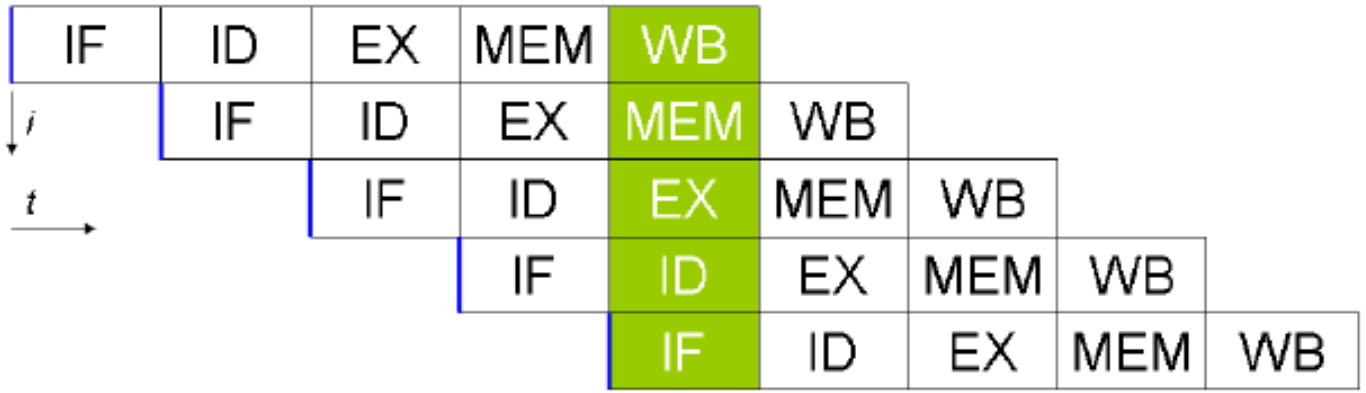

MEMORIA
REGISTROS DE LA CPU
300
1940
301
5941
302
2941
302
PC
0005
AC
2941
IR
940
0003
941
0002
PASO 5
MEMORIA
REGISTROS DE LA CPU
300
1940
301
5941
302
2941
302
PC
0005
AC
2941
IR
940
0003
941
0005
PASO 6

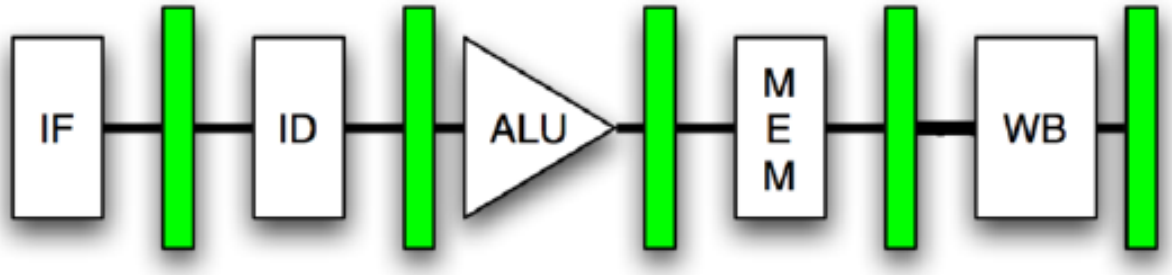
ESTADO PIPELINE

Técnica de implementación de paralelismo a nivel de instrucción en un único procesador. Se busca que todas las unidades del procesador estén ocupadas durante el ciclo de instrucción.

Cada unidad ejecuta una tarea del ciclo de instrucción de instrucciones diferentes, aumentando el rendimiento del sistema al ser posible elevar la frecuencia del reloj de la CPU.

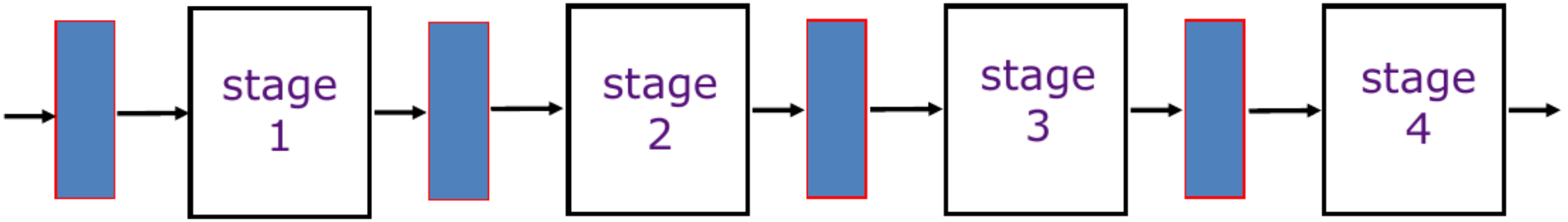
ESTADO PIPELINE





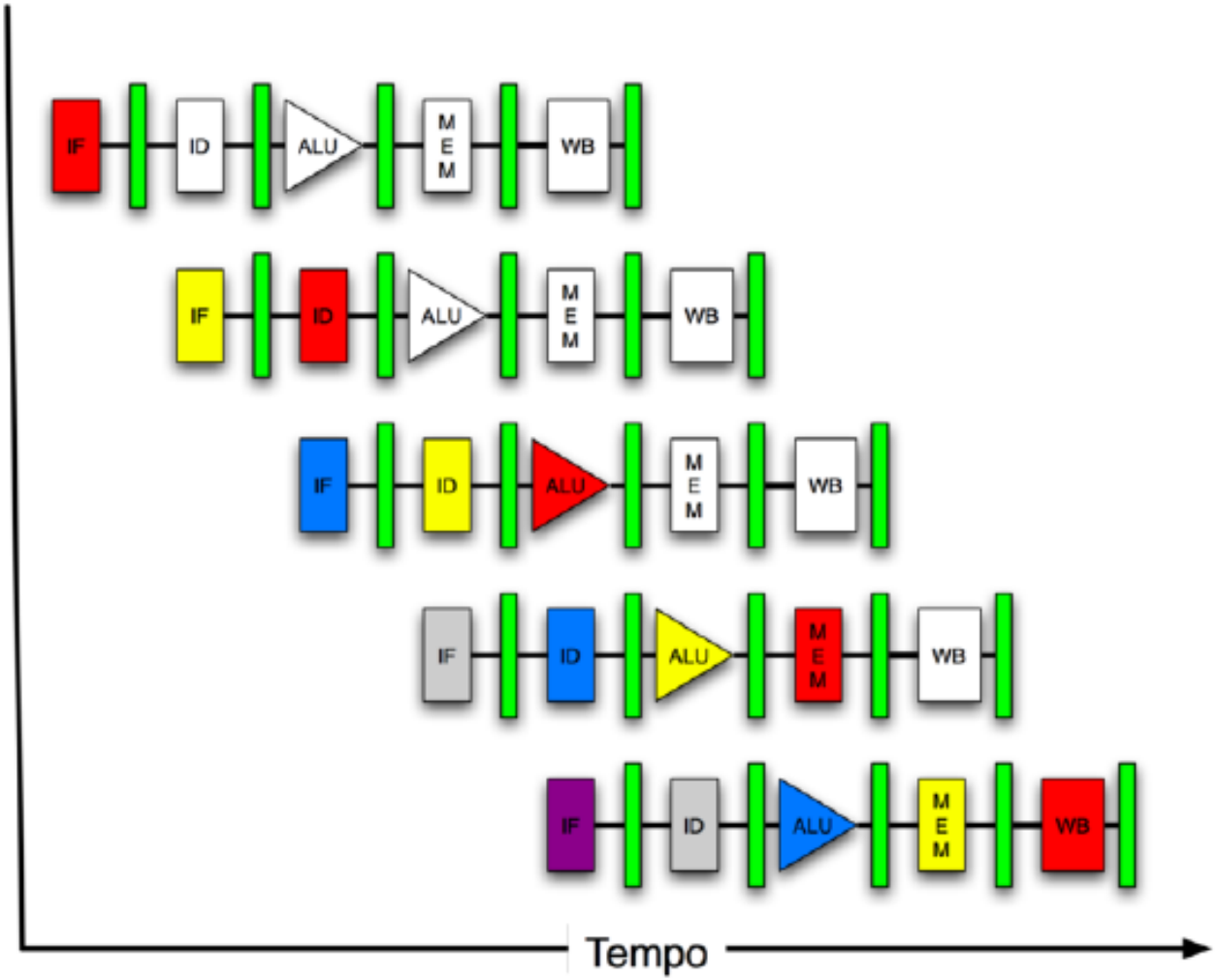
Los registros deben mantener los datos a la entrada de cada unidad del procesador para garantizar la ejecución de cada instrucción.

ESTADO PIPELINE IDEAL

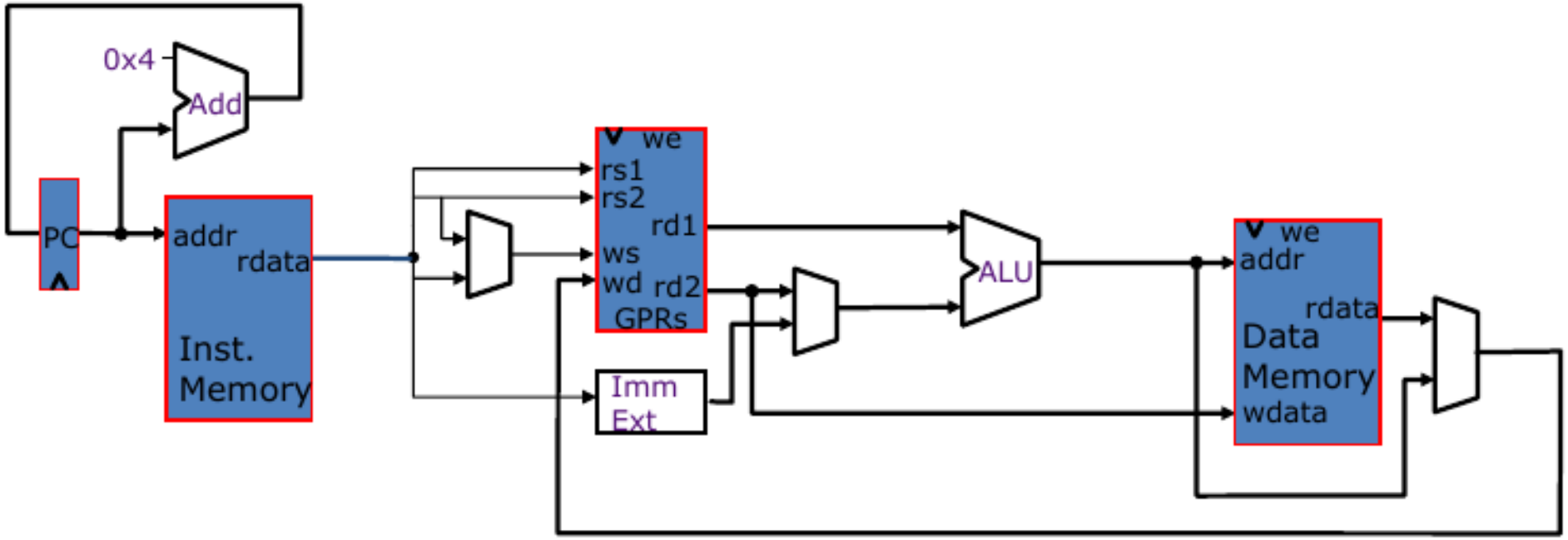


- Todas las instrucciones pasan por los mismos estados.
- No se comparten recursos entre dos etapas.
- Los retardos de propagación es igual en todos los estados.
- La transición de un estado que entra de una instrucción no se afecta por las transiciones de otras etapas.

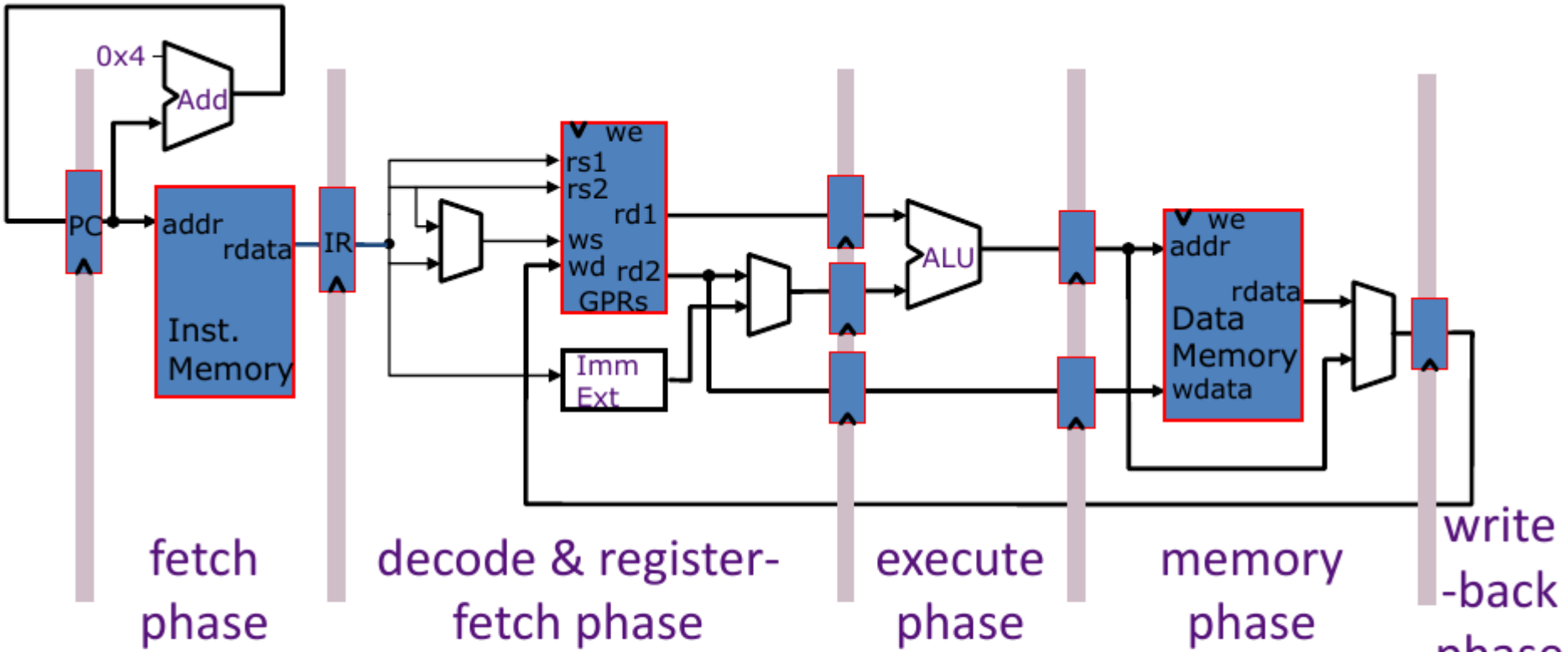
FILOSOFÍA PIPELINE



ESTADO PIPELINE SIMPLIFICADO



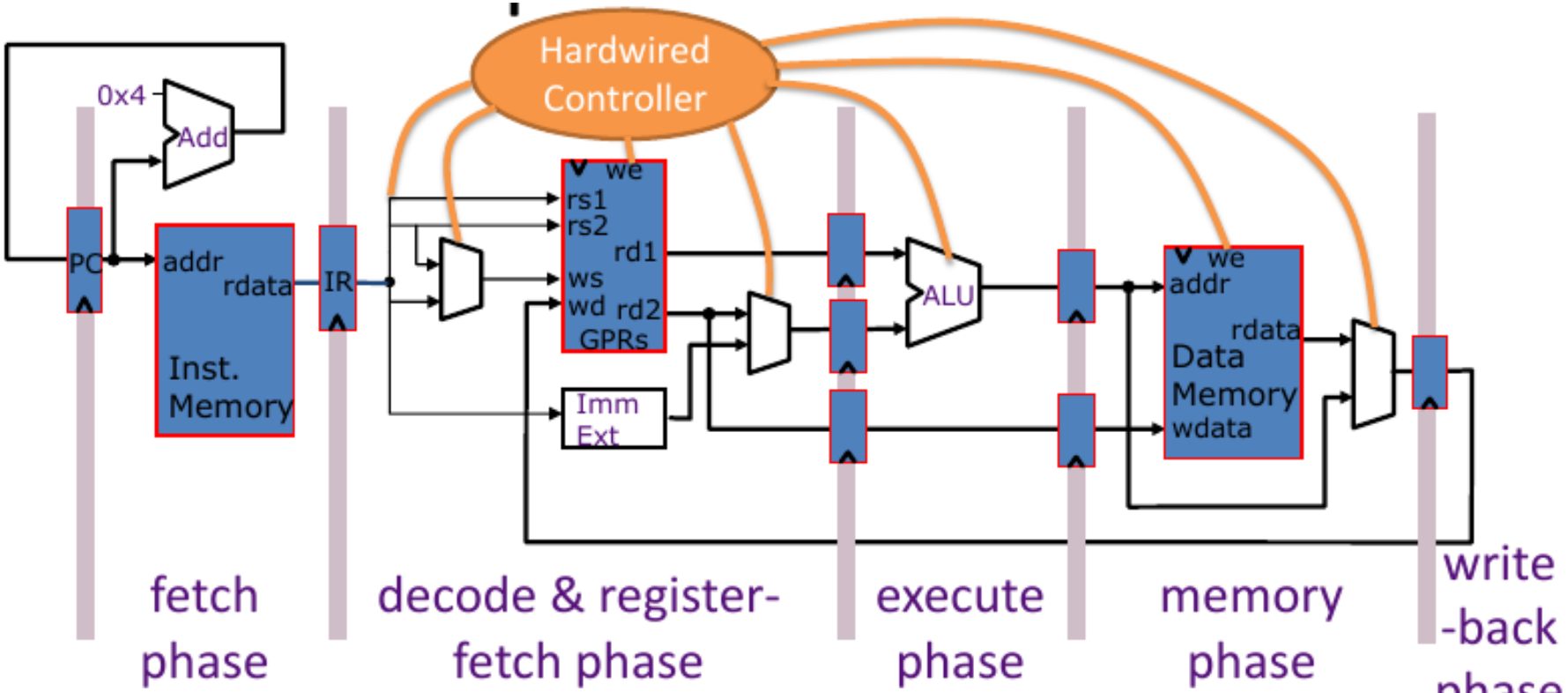
ESTADO PIPELINE



Clock period can be reduced by dividing the execution of an instruction into multiple cycles

$$t_C > \max\{t_{IM}, t_{RF}, t_{ALU}, t_{DM}, t_{RW}\} \; (= t_{DM} \; \textit{probably})$$

ESTADO PIPELINE.



Clock period can be reduced by dividing the execution of an instruction into multiple cycles

$$t_C > \max\{t_{IM}, t_{RF}, t_{ALU}, t_{DM}, t_{RW}\} \; (= t_{DM} \textit{ probably})$$

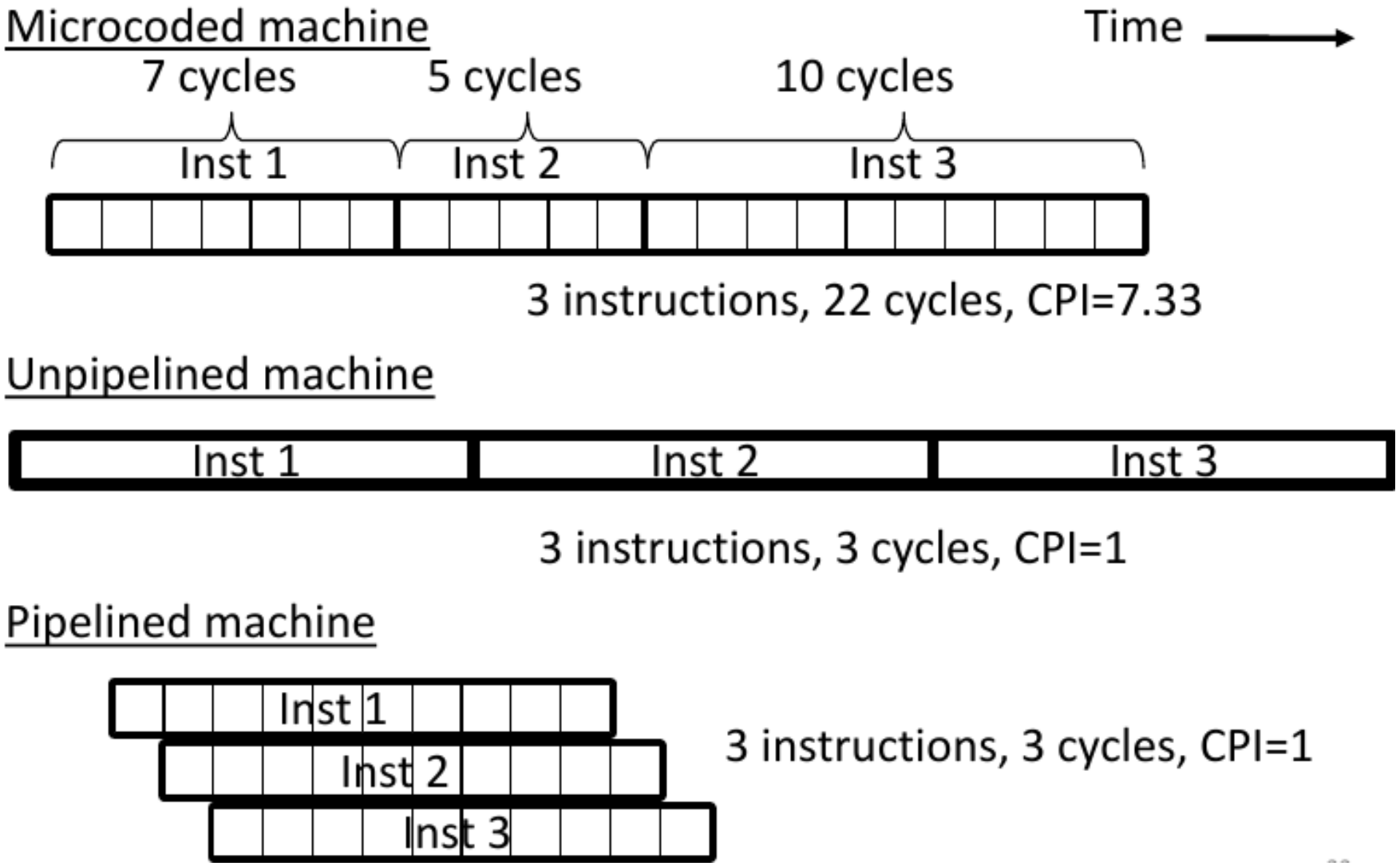
LEY DEL RENDIMIENTO DEL PROCESADOR

$$\frac{Tiempo}{Programa} = \frac{Instrucciones}{Programa} * \frac{Ciclo}{Instrucción} * \frac{Tiempo}{Ciclo}$$

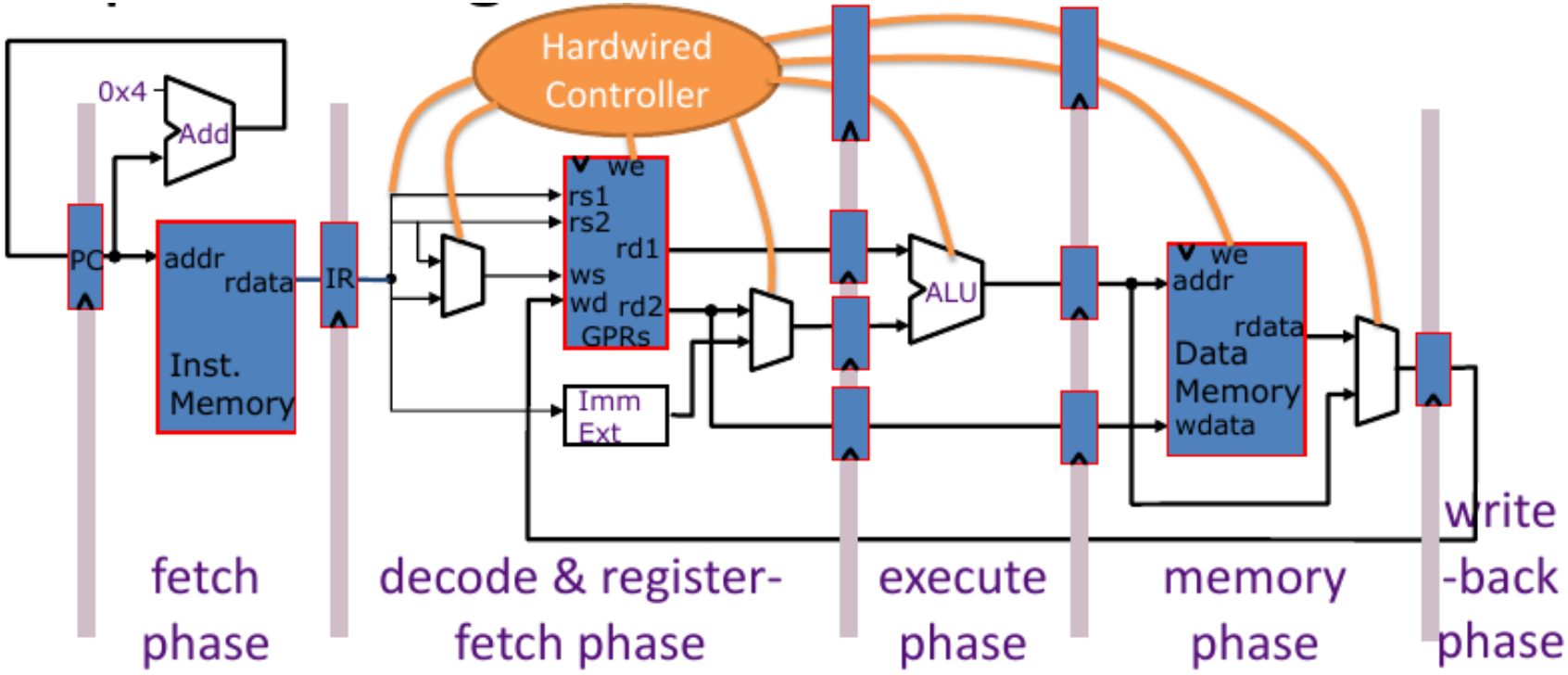
- Instrucciones por programa, depende del código fuente, la tecnología del compilador.
- Ciclo por instrucción CPI depende del ISA (instruction set architecture).
- El tiempo por instrucción depende de la microarquitectura y la tecnología de fabricación.

Microarchitecture	CPI	cycle time
Microcoded	>1	short
Single-cycle unpipelined	1	long
Pipelined	1	short
Multi-cycle, unpipelined control	>1	short

EJEMPLO DE CPI



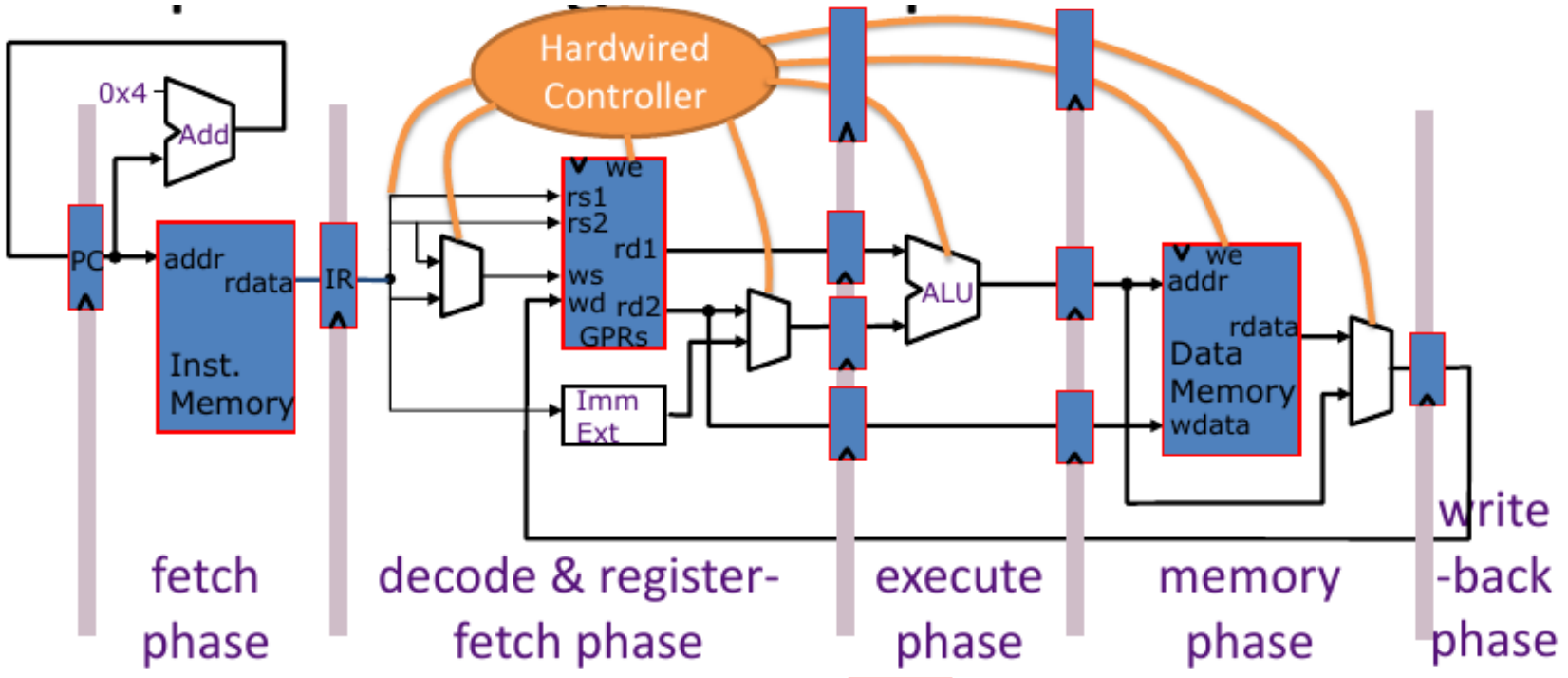
ESTADO PIPELINE: TRANSICIONES Vs TIME.



time	t0	t1	t2	t3	t4	t5	t6	t7	. . . .
instruction1	IF_1	ID_1	EX_1	MA_1	WB_1				
instruction2		IF_2	ID_2	EX_2	MA_2	WB_2			
instruction3			IF_3	ID_3	EX_3	MA_3	WB_3		
instruction4				IF_4	ID_4	EX_4	MA_4	WB_4	
instruction5					IF_5	ID_5	EX_5	MA_5	WB_5

25

ESTADO PIPELINE: ESTADO Vs TIME.



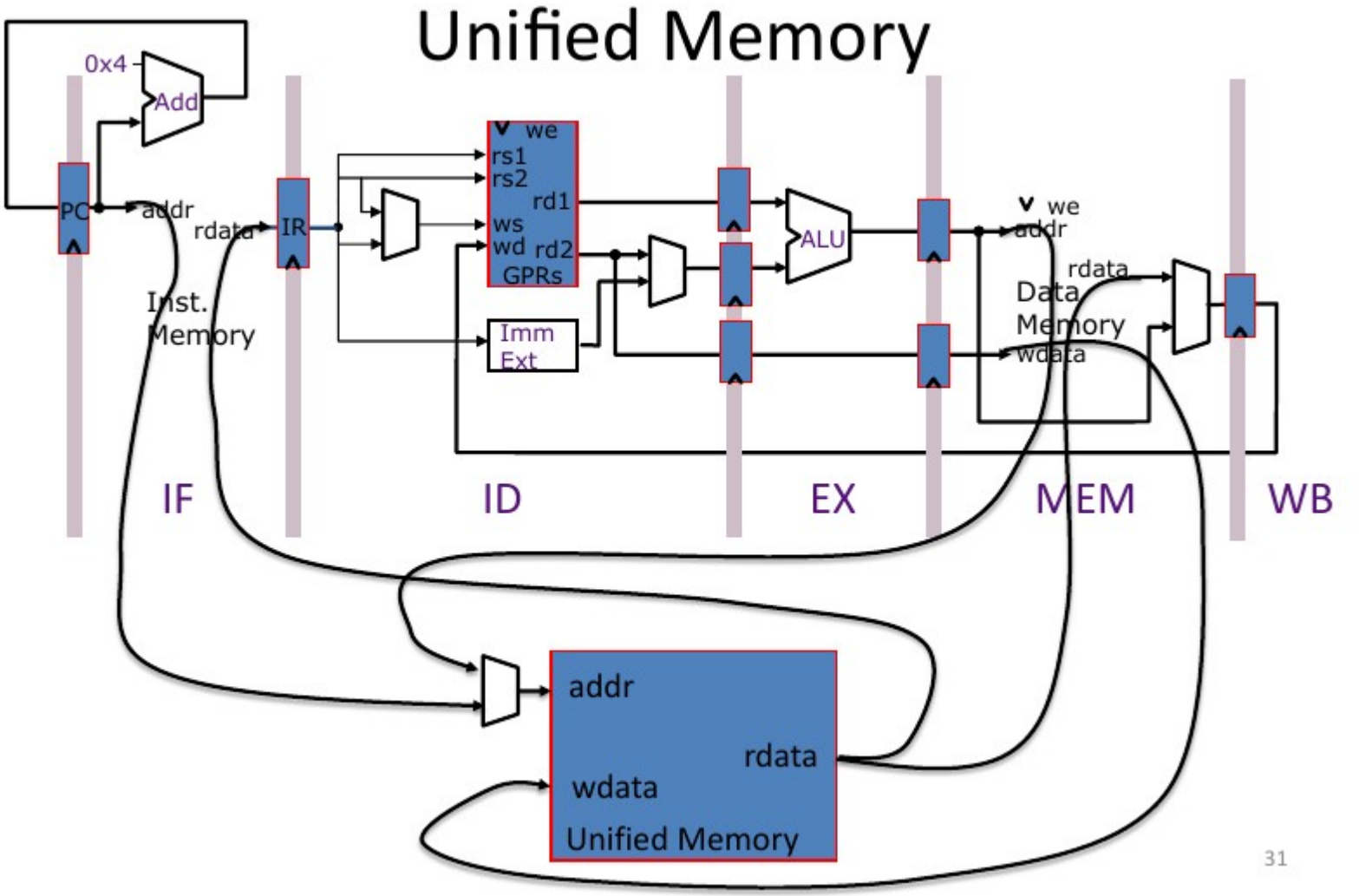
Resources	time	t0	t1	t2	t3	t4	t5	t6	t7	. . . .
	IF	I_1	I_2	I_3	I_4	I_5				
	ID		I_1	I_2	I_3	I_4	I_5			
	EX			I_1	I_2	I_3	I_4	I_5		
	MA				I_1	I_2	I_3	I_4	I_5	
	WB					I_1	I_2	I_3	I_4	I_5

26

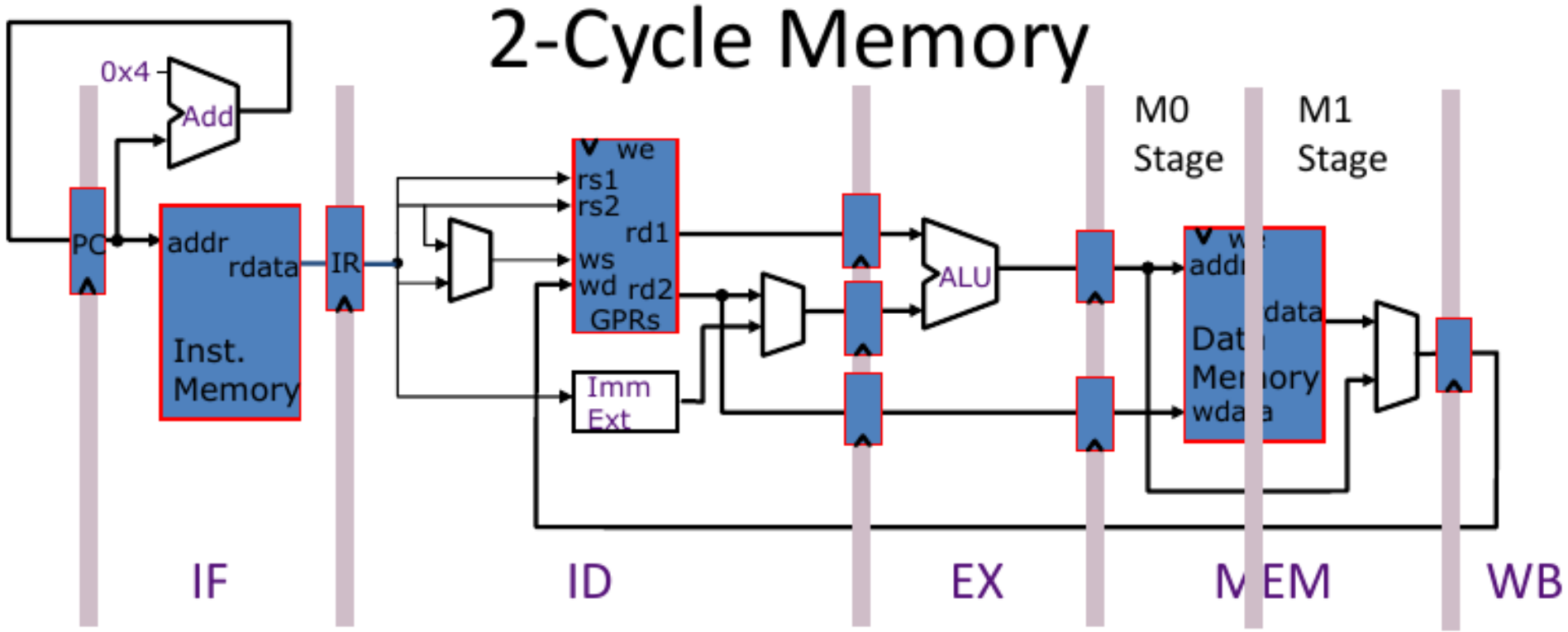
PROBLEMAS DEL ESTADO PIPELINE.

- ESTRUCTURALES: DOS INSTRUCCIONES DIFERENTES RECURREN A LOS MISMOS RECURSOS DE HARDWARE EN LOS MISMOS ESTADO DE TIEMPO.
- DATOS: UNA INSTRUCCIÓN DEPENDE DE DATOS GENERADOS EN LA EJECUCIÓN DE LA INSTRUCCIÓN ANTERIOR DEL PIPELINE.
- CONTROL: EL ESTADO PIPELINE SE INTERRUMPE POR LA NO SECUENCIA DEL PROGRAMA Y DEBE REINICIARCE.

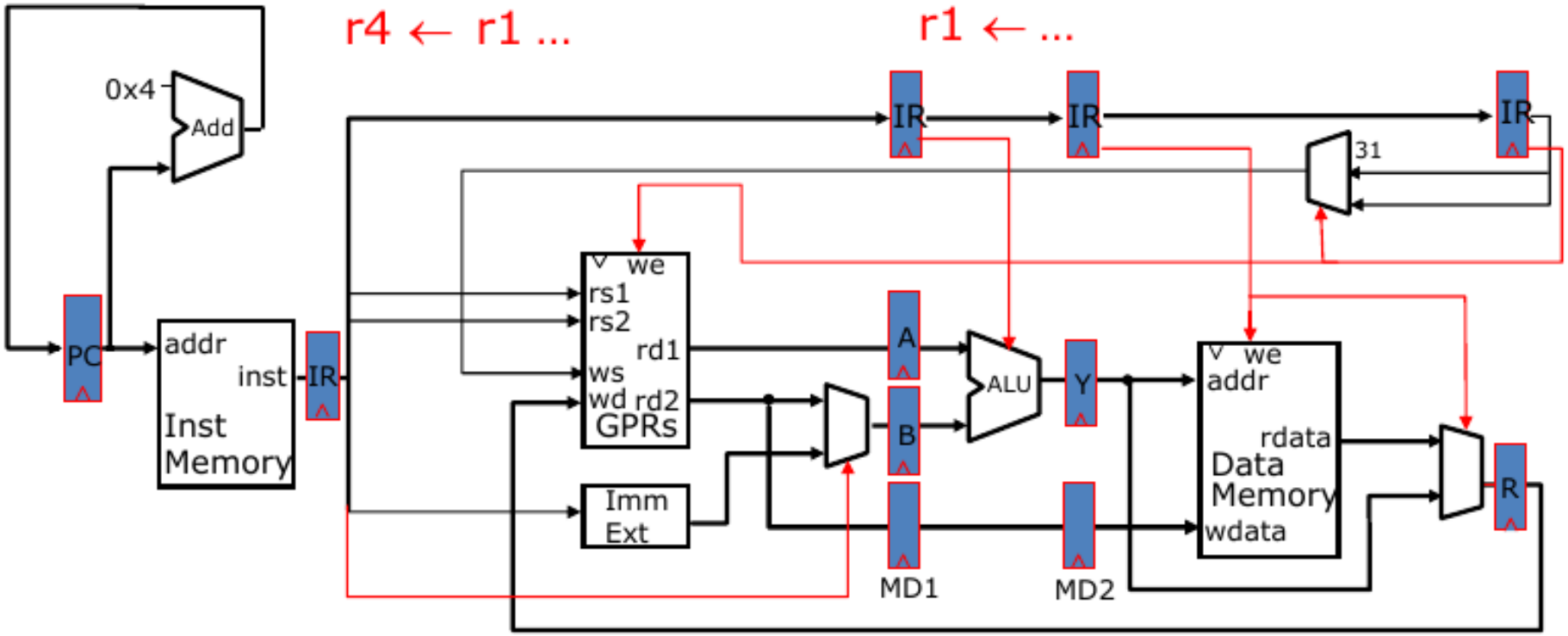
PROBLEMAS DEL ESTADO PIPELINE: ESTRUCTURAL



PROBLEMAS DEL ESTADO PIPELINE: ESTRUCTURAL



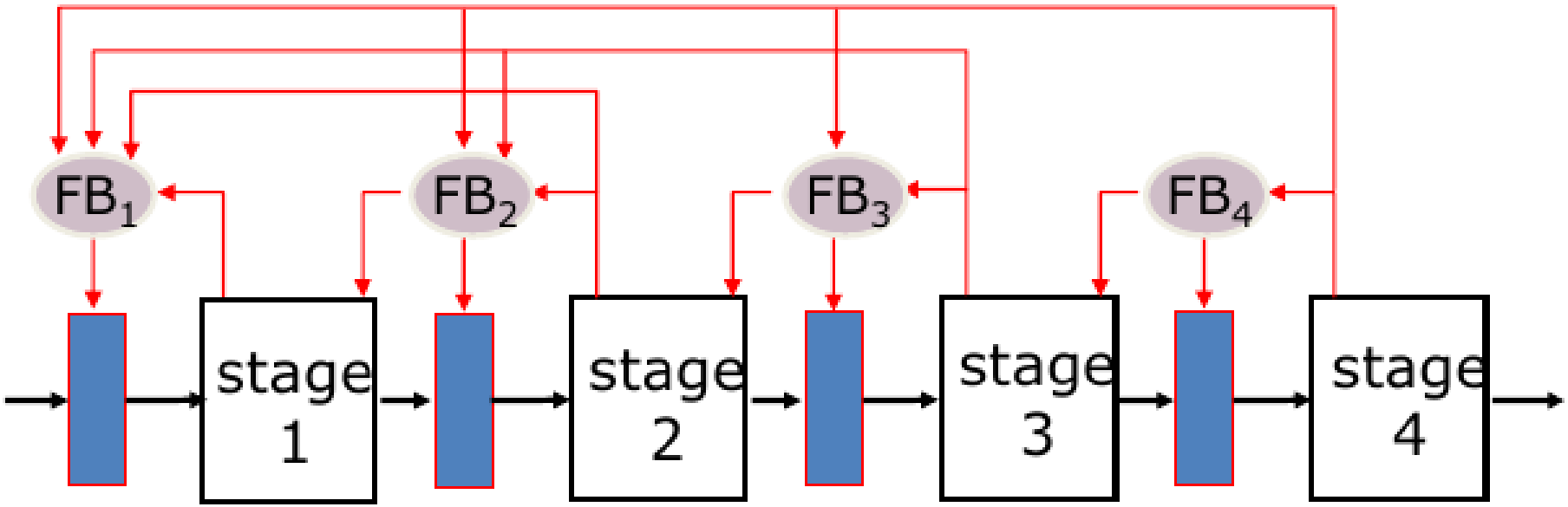
PROBLEMAS DEL ESTADO PIPELINE: DATOS



...
r1 ← r0 + 10 (ADDI R1, R0, #10)
r4 ← r1 + 17 (ADDI R4, R1, #17)
...

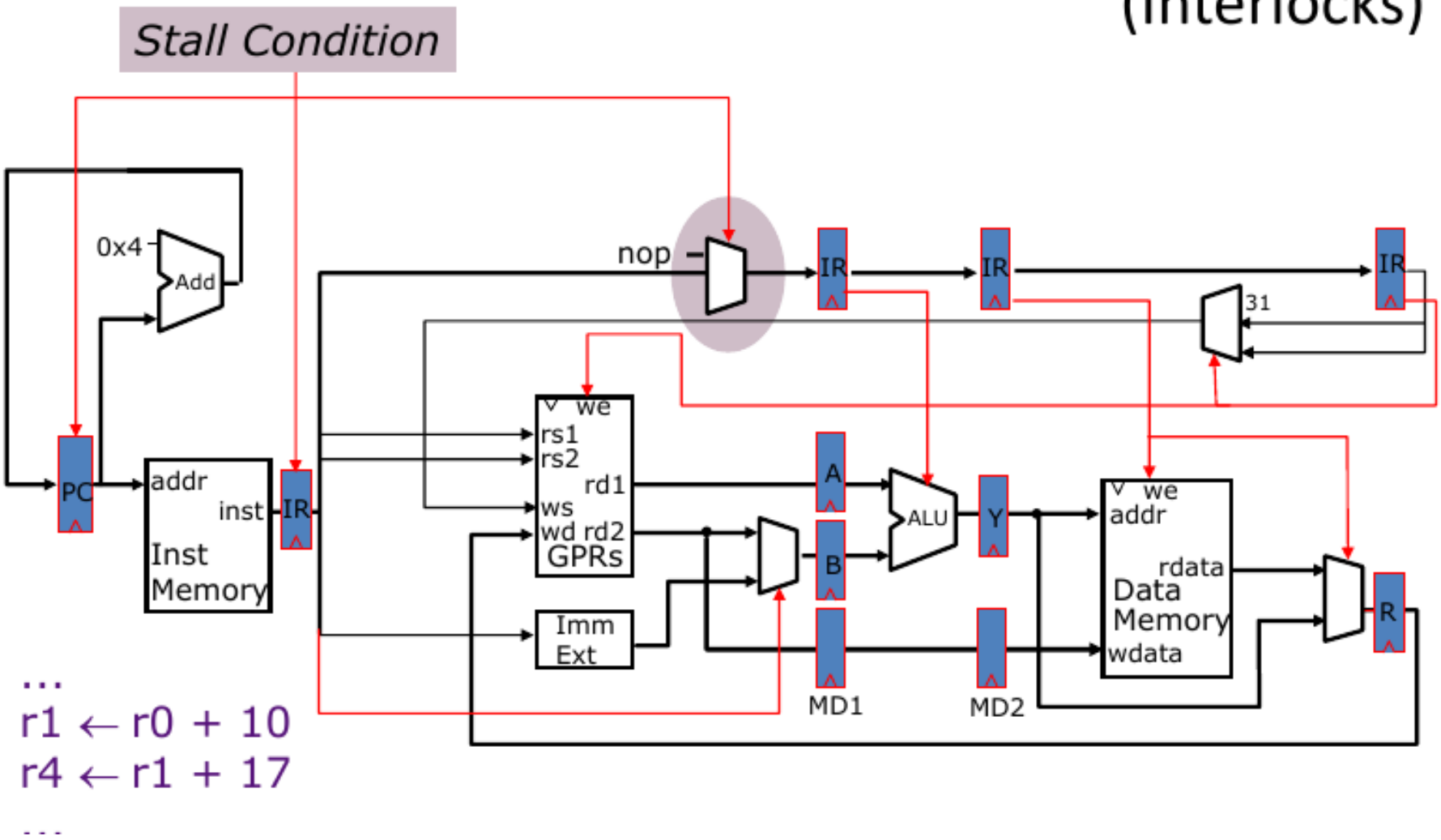
r1 is stale. Oops!

PROBLEMAS DEL ESTADO PIPELINE: RETRO ALIMENTACIÓN

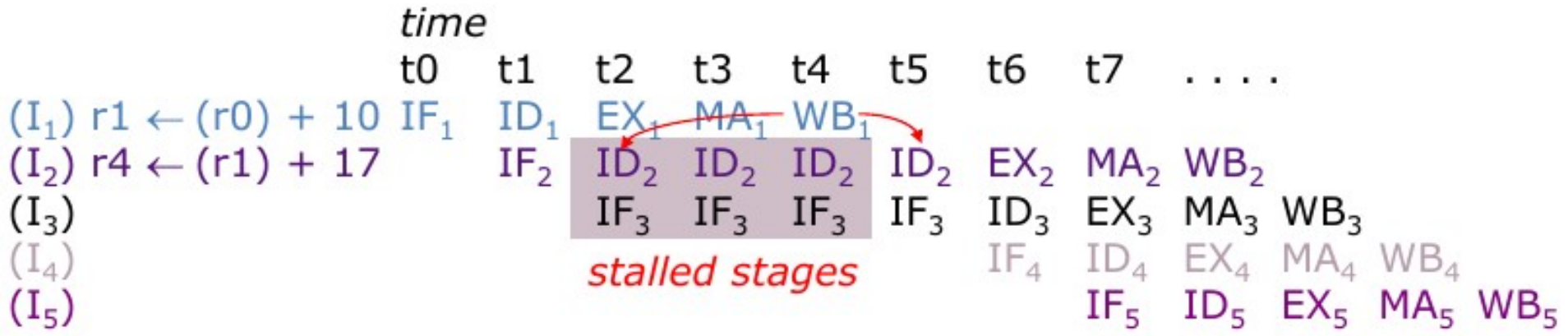


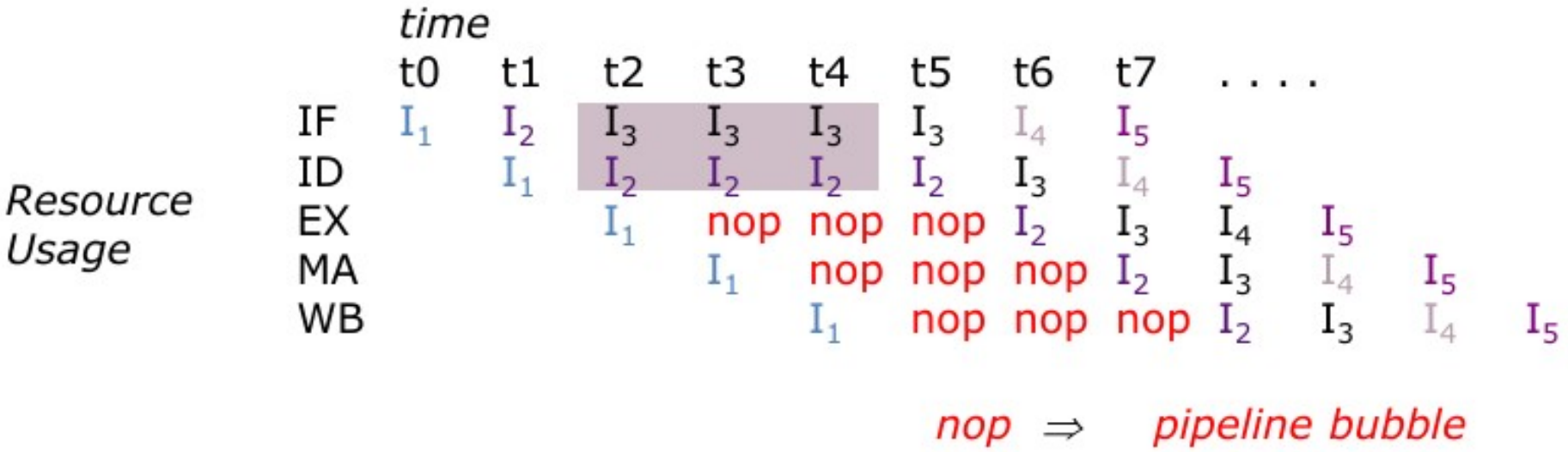
PROBLEMA DE ESTADO PIPELINE: RETRO ALIMENTACIÓN

(Interlocks)

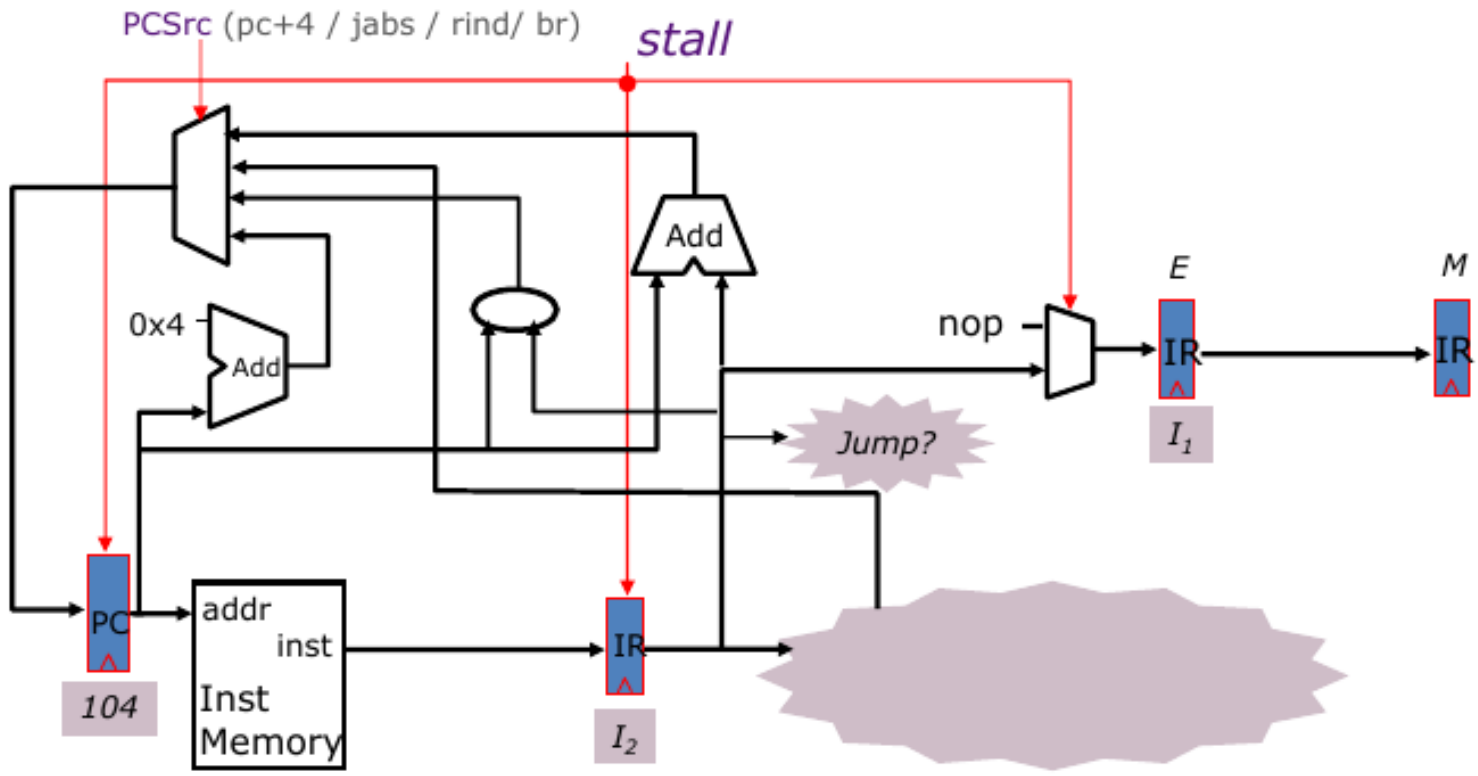


PROBLEMAS DE ESTADO PIPELINE: RETRO ALIMENTACIÓN





PROBLEMA ESTADO PIPELINE: CONTROL



I_1	096	ADD
I_2	100	J 304
I_3	~~104~~	~~ADD~~
I_4	304	ADD

kill

A jump instruction kills (not stalls) the following instruction

How?

PROBLEMAS ESTADO PIPELINE: CONTROL

	time t0	t1	t2	t3	t4	t5	t6	t7	. . . .
(I_1) 096: ADD	IF_1	ID_1	EX_1	MA_1	WB_1				
(I_2) 100: J 304		IF_2	ID_2	EX_2	MA_2	WB_2			
(I_3) 104: ADD			IF_3	nop	nop	nop	nop		
(I_4) 304: ADD				IF_4	ID_4	EX_4	MA_4	WB_4	

Resource Usage

	time t0	t1	t2	t3	t4	t5	t6	t7	. . . .
IF	I_1	I_2	I_3	I_4	I_5				
ID		I_1	I_2	nop	I_4	I_5			
EX			I_1	I_2	nop	I_4	I_5		
MA				I_1	I_2	nop	I_4	I_5	
WB					I_1	I_2	nop	I_4	I_5

nop ⇒ *pipeline bubble*

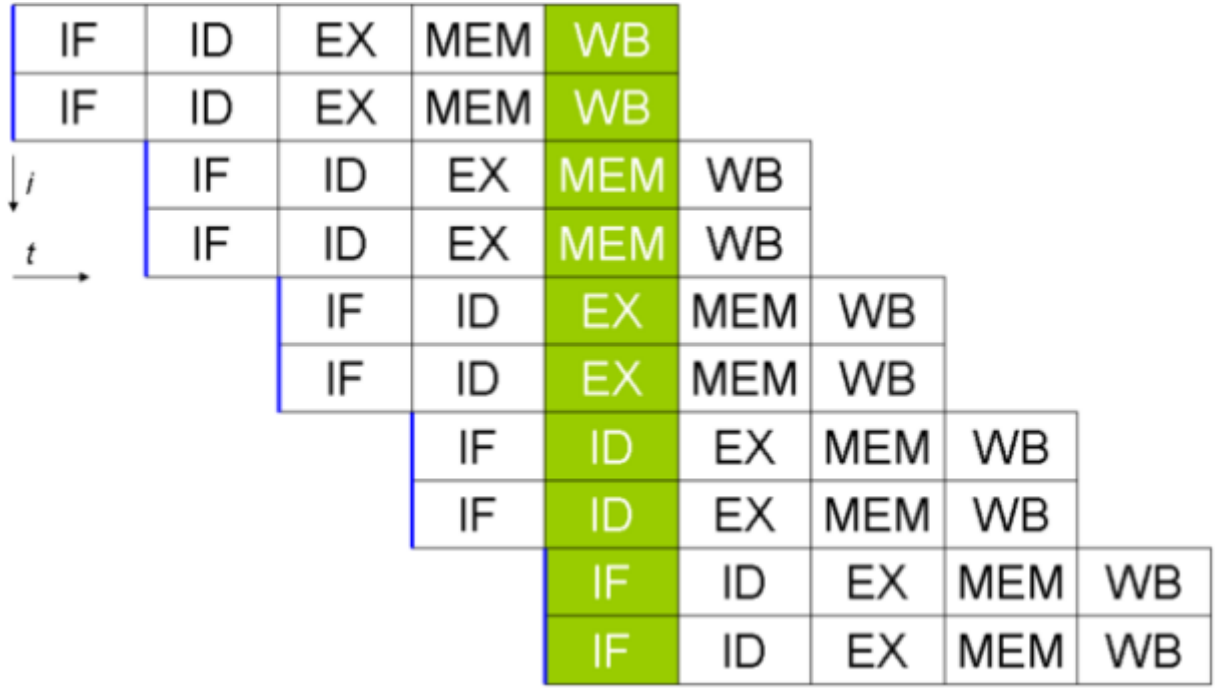
SUPER PIPELINE

Es la técnica de elevar la profundidad del paralelismo de las instrucciones del pipeline para aumentar la velocidad del reloj y reducir la latencia de las etapas individuales.

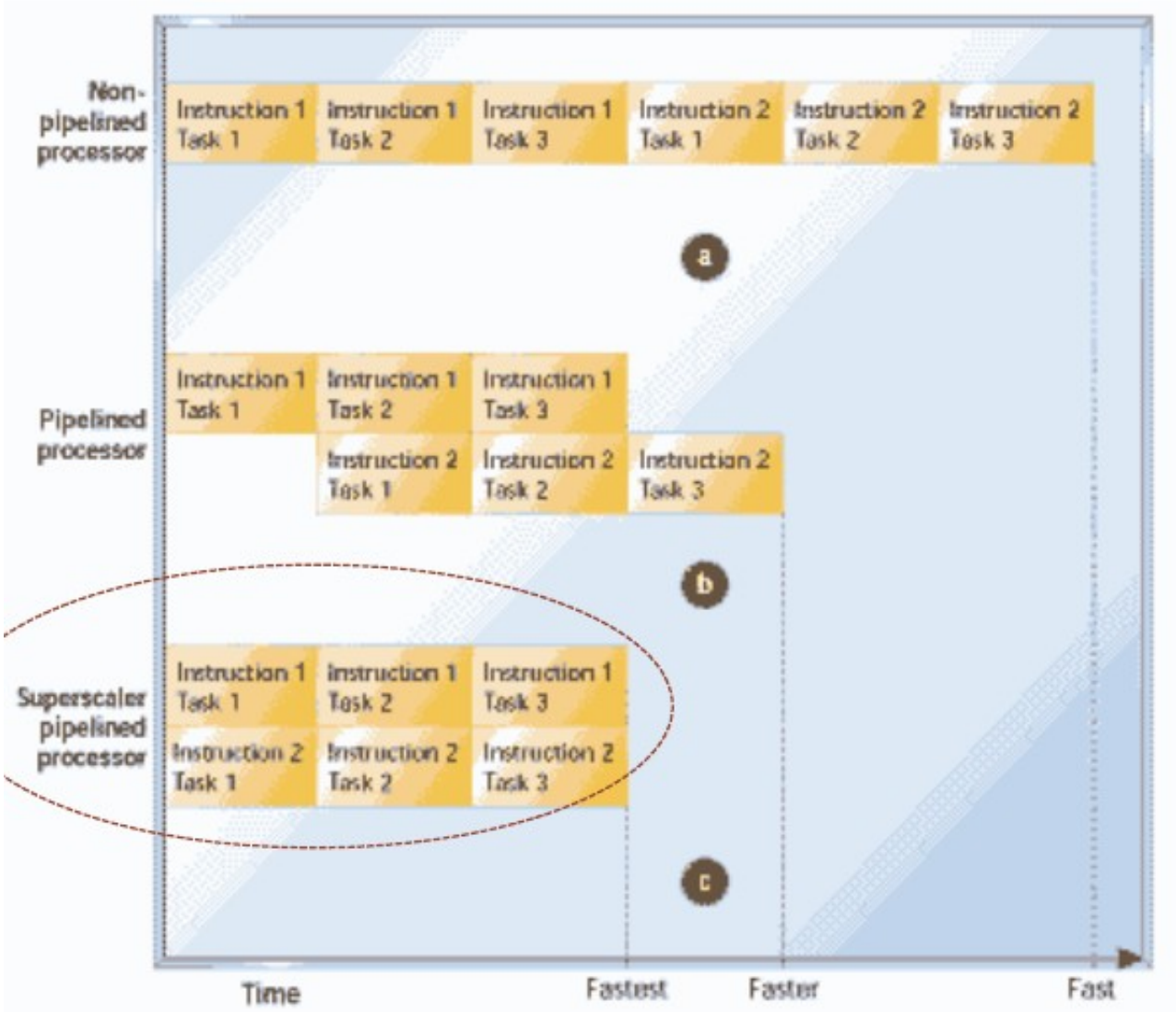
Si la ALU toma tres veces más tiempo que cualquier otro módulo, podemos dividir la ALU en tres etapas separadas, lo que reducirá la cantidad de tiempo perdido en etapas más cortas. El problema aquí es que necesitamos encontrar una manera de subdividir nuestras etapas en etapas más cortas, y también necesitamos construir unidades de control más complicadas para operar la tubería y prevenir todos los posibles peligros.

SUPERESCALER

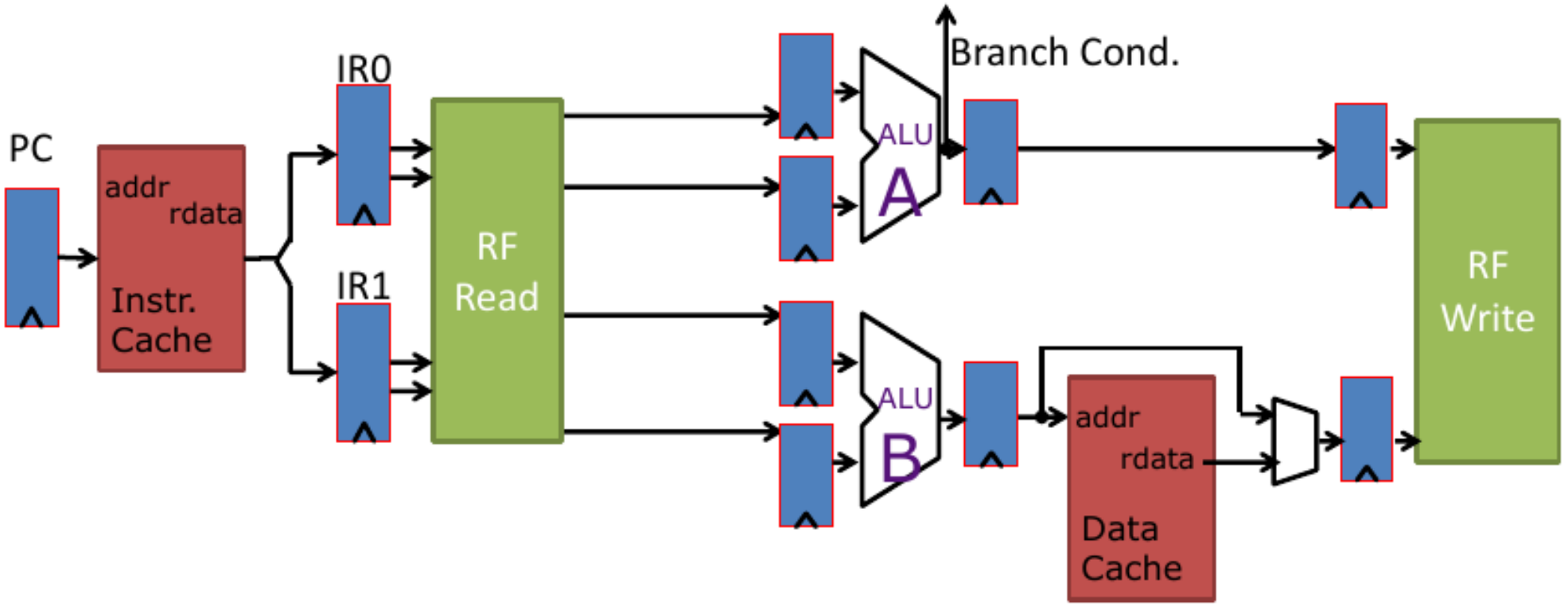
Arquitectura en la que un procesador es capaz de ejecutar mas de una instrucción por ciclo de reloj. En esta arquitectura se replica las unidades para lograr ejecutar múltiples instrucciones.



SUPERESCALER



SUPERESCALAR



SUPERESCALAR