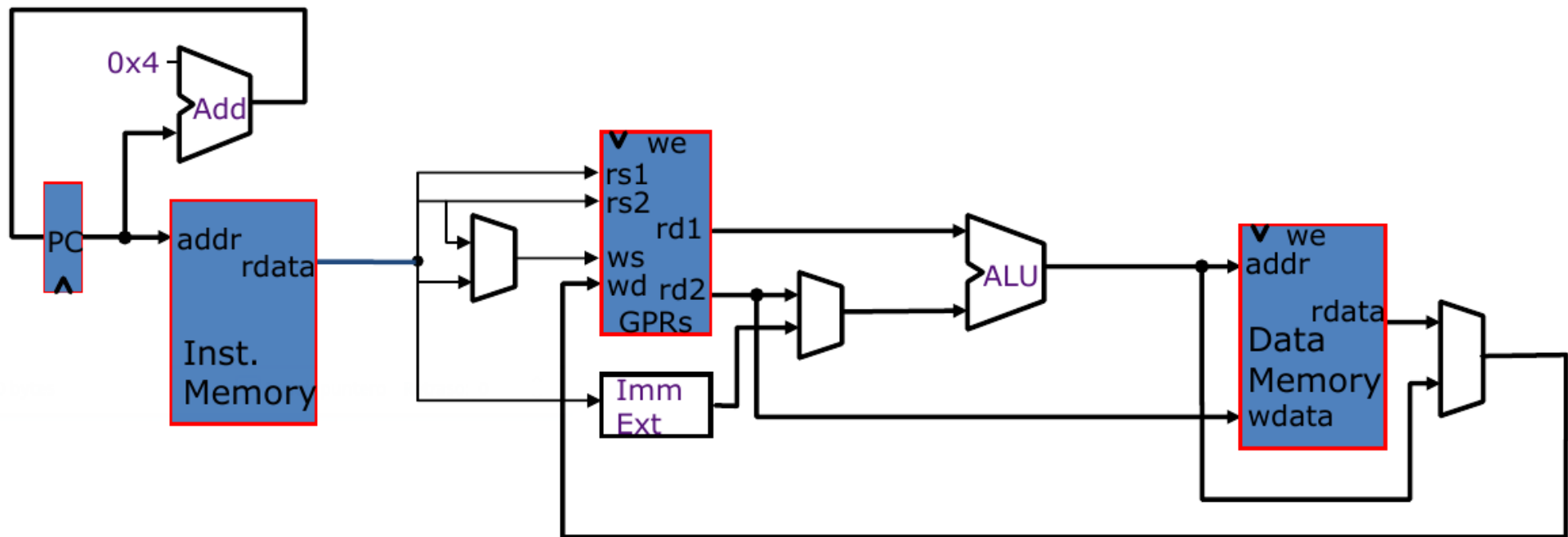


ARQUITECTURA DE COMPUTADORES

CICLO DE INSTRUCCIÓN

Ing Marlon Moreno Rincón

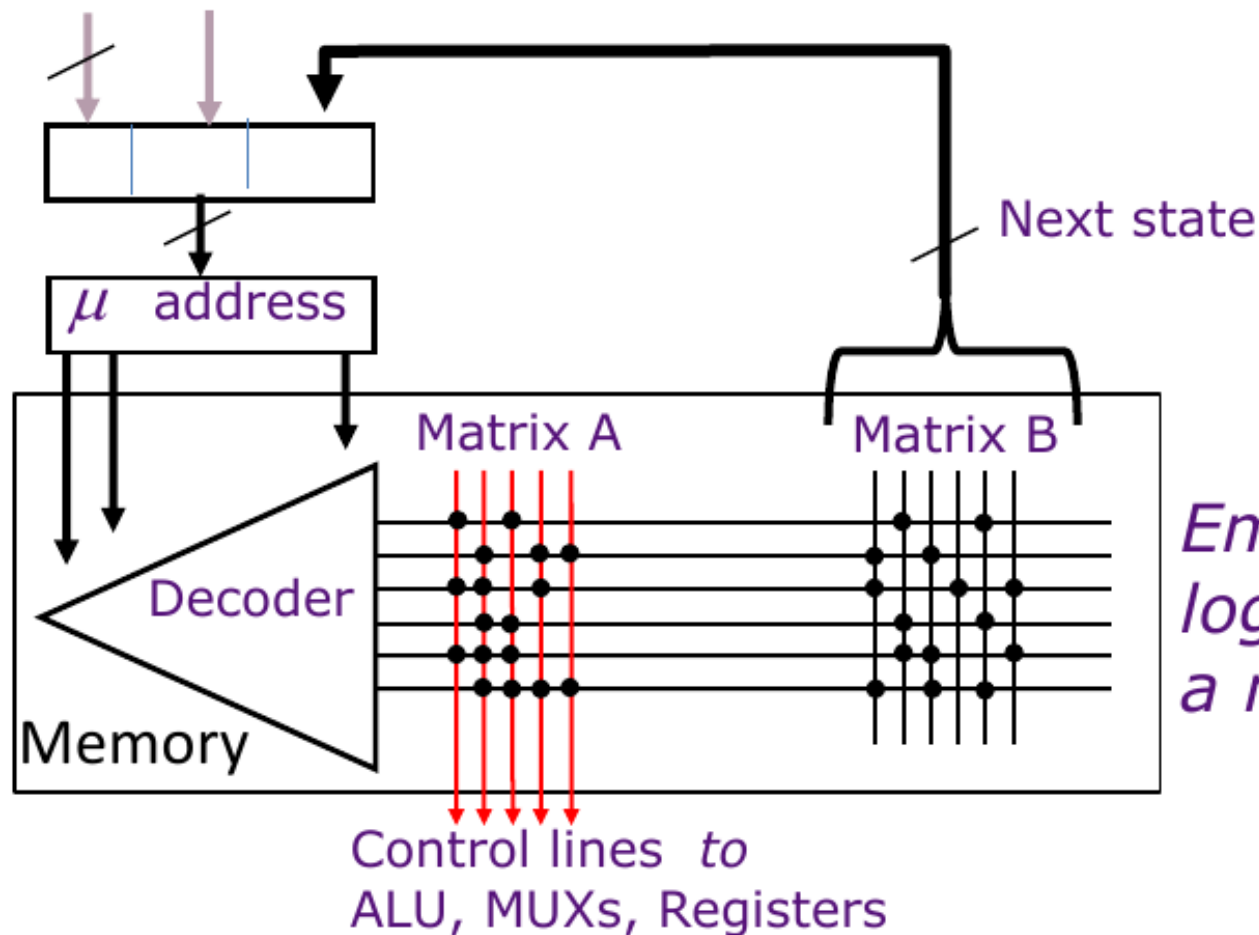
FLUJO DE DATOS SIMPLIFICADA.



UNIDAD DE CONTROL CABLEADA.

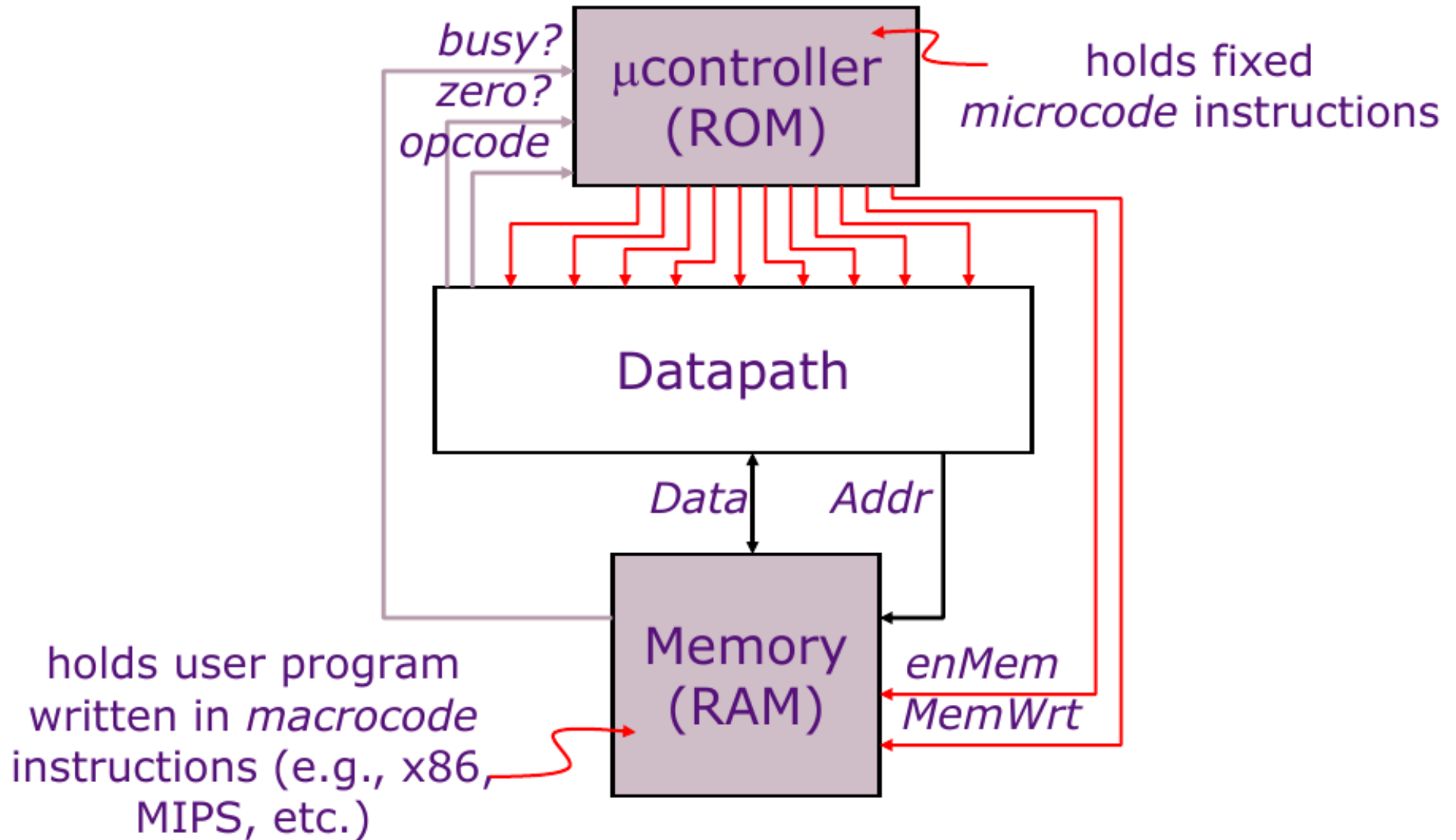
op code conditional
code flip-flop

*First used in EDSAC-2,
completed 1958*



*Embed the control
logic state table in
a memory array*

UNIDAD DE CONTROL MICROPROGRAMADA.



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.

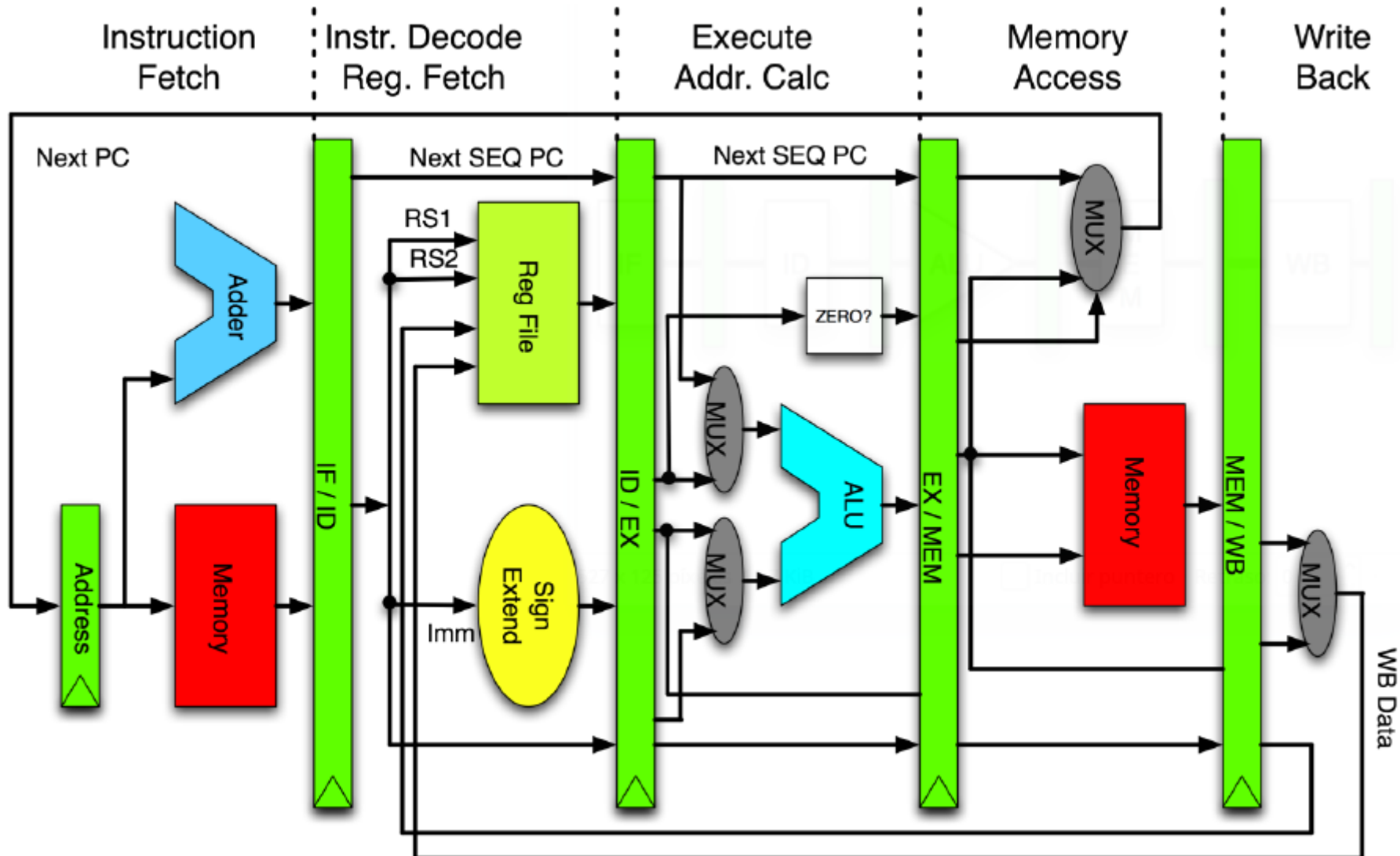
Ciclo maquina: Periodo mínimo que tarda la unidad central de procesamiento en ejecutar una instrucción. Se toma la instrucción que requiere menos tiempo.

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 guarda 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 disecione otra interrupción. La unidad de control decodifica la operación a realizar y mueve los datos de memoria a la unidad de procesamiento.

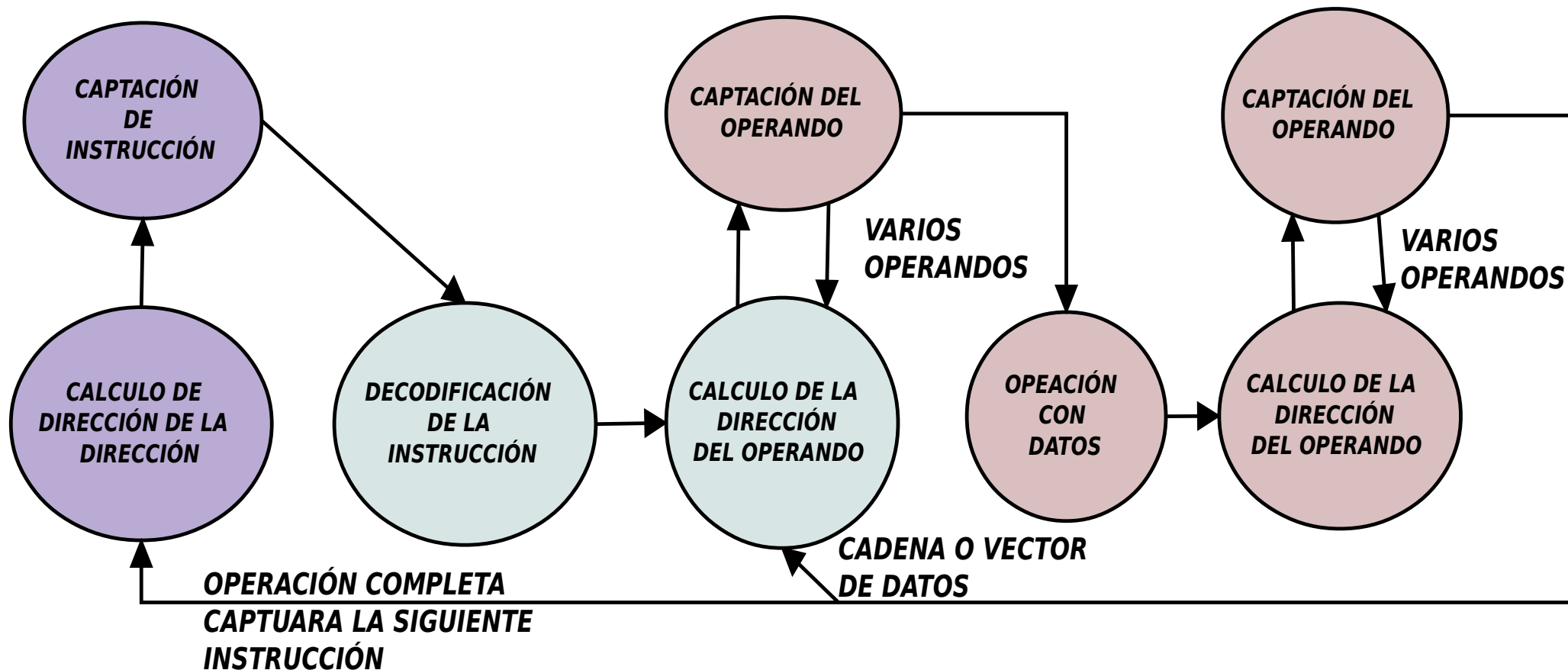
CICLO DE INSTRUCCIÓN – EXECUTION

Ejecución de la instrucción: La unidad de control envía 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 resultados son almacenados en memoria o enviados a los diferentes dispositivos. El contador de programa se incrementa o se modifica de acuerdo a los resultados obtenidos de la operación.

CICLO DE INSTRUCCIÓN

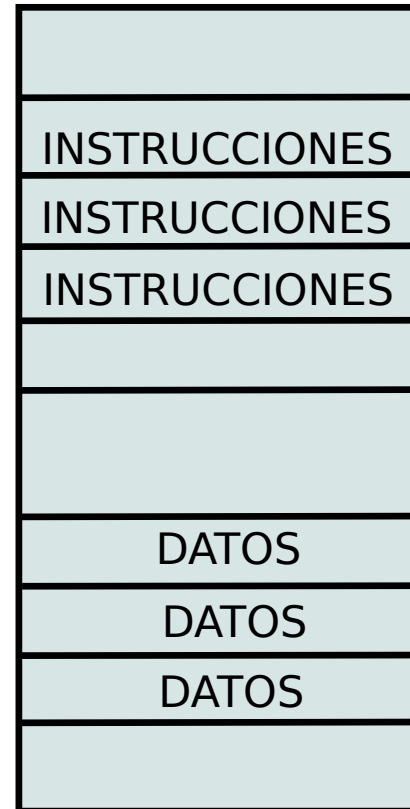
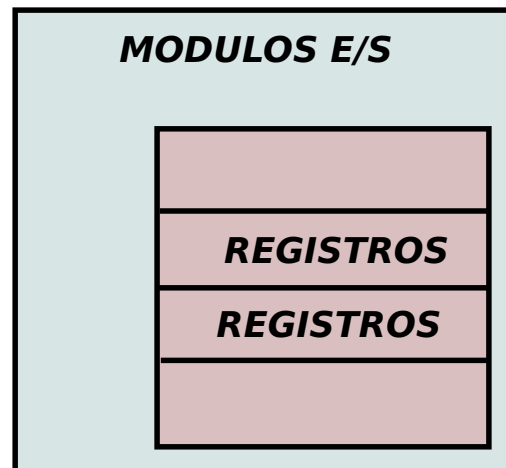
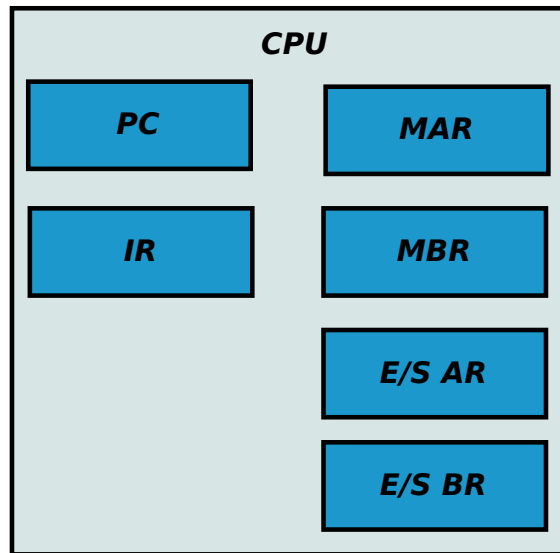


EJEMPLO

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 las diapositivas 14 y 15 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

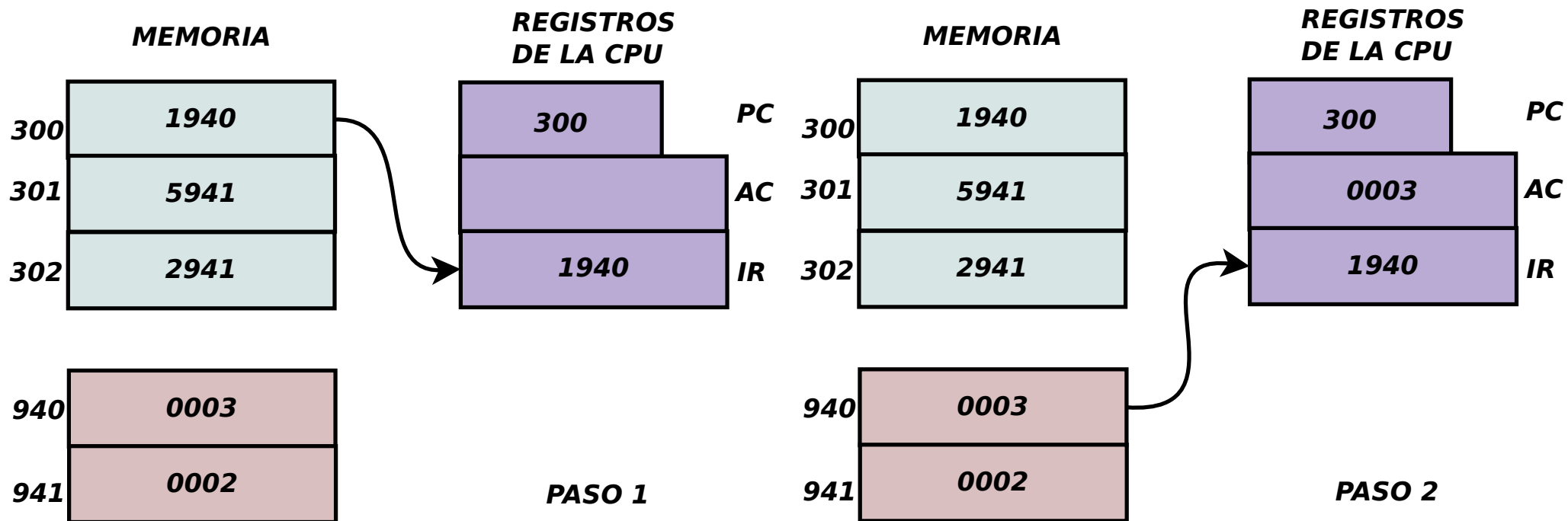


Formato de datos

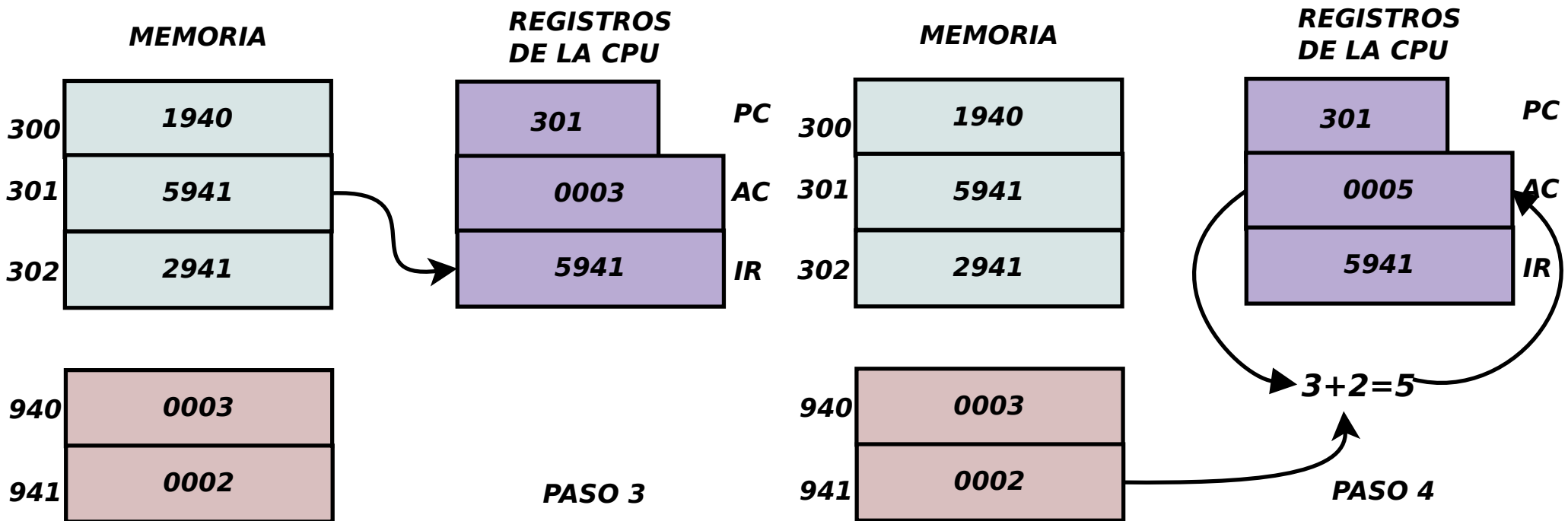
Lista parcial de código operación

- 0001 = Cargar AC desde memoria**
- 0010 = Almacenar AC en memoria**
- 0101 = Sumar a AC un dato de memoria**

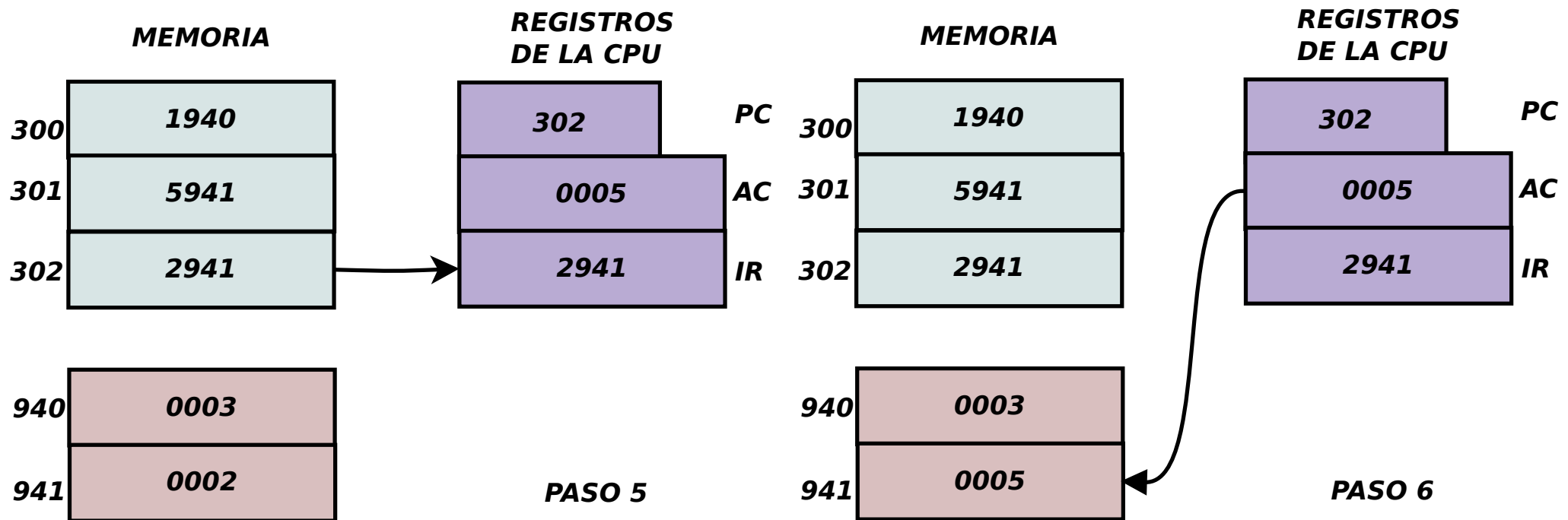
EJEMPLO



EJEMPLO



EJEMPLO

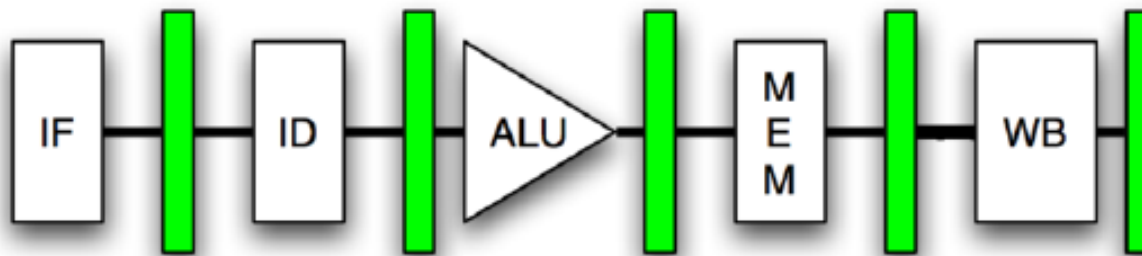


ESTADO PIPELINE

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.

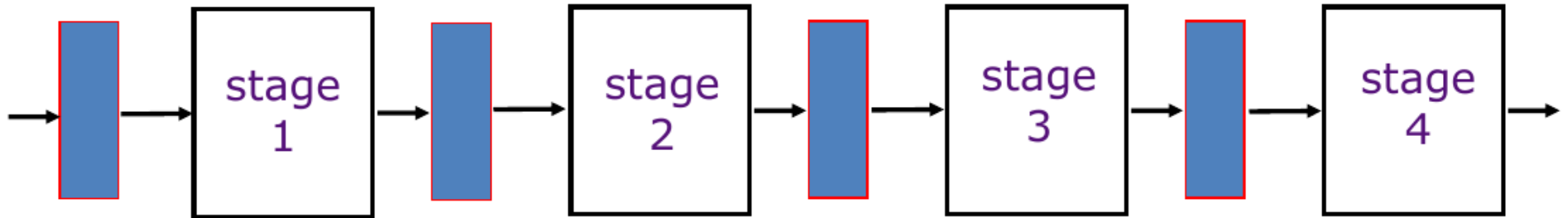
Cada unidad ejecuta una tarea del ciclo de instrucción de instrucciones diferentes, aumentando el rendimiento del sistema al ser posible elevar la frecuencia del reloj de la CPU.

ESTADO PIPELINE



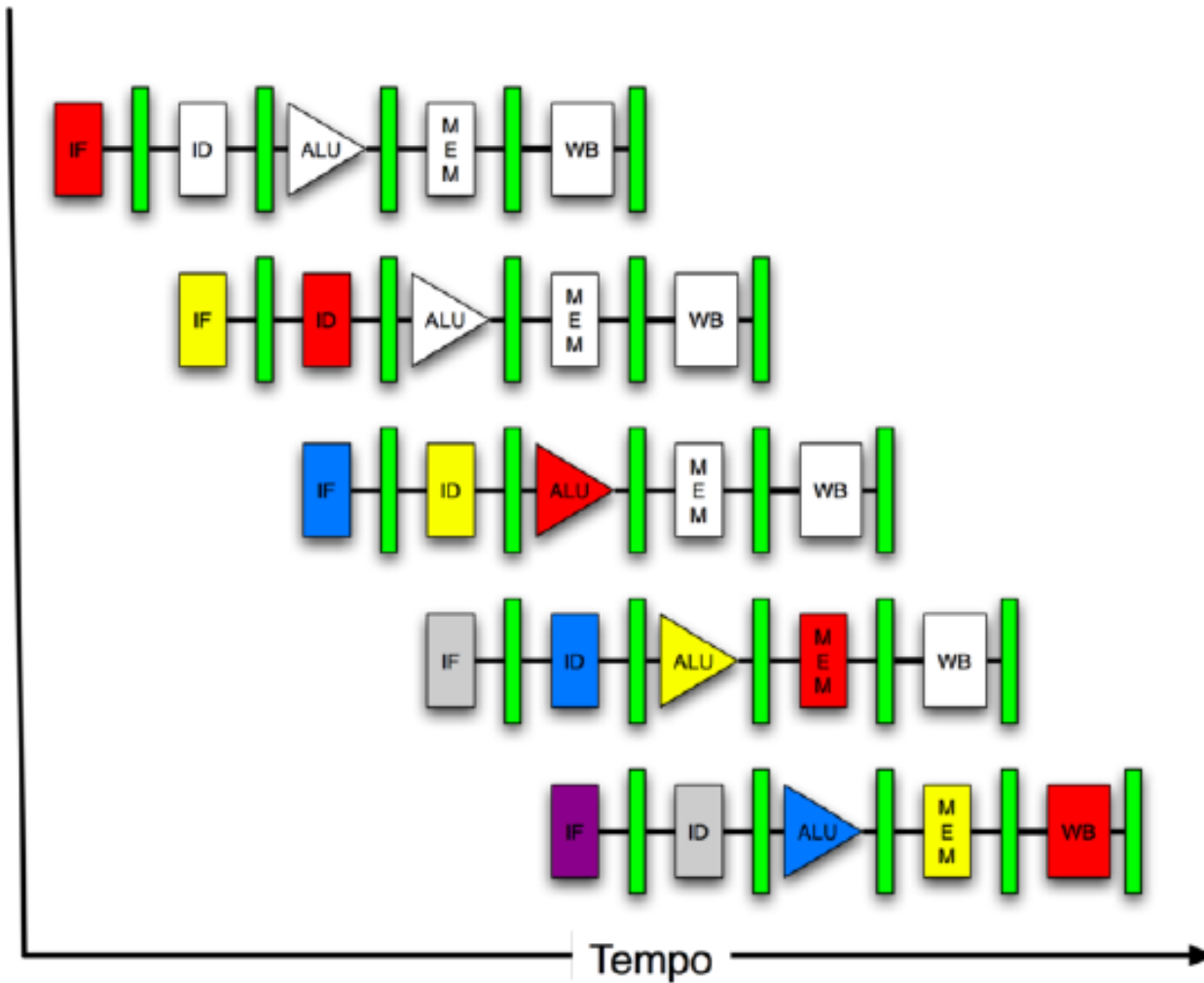
Los registros deben mantener los datos a la entrada de cada unidad del procesador para garantizar la ejecución de cada instrucción.

ESTADO PIPELINE IDEAL

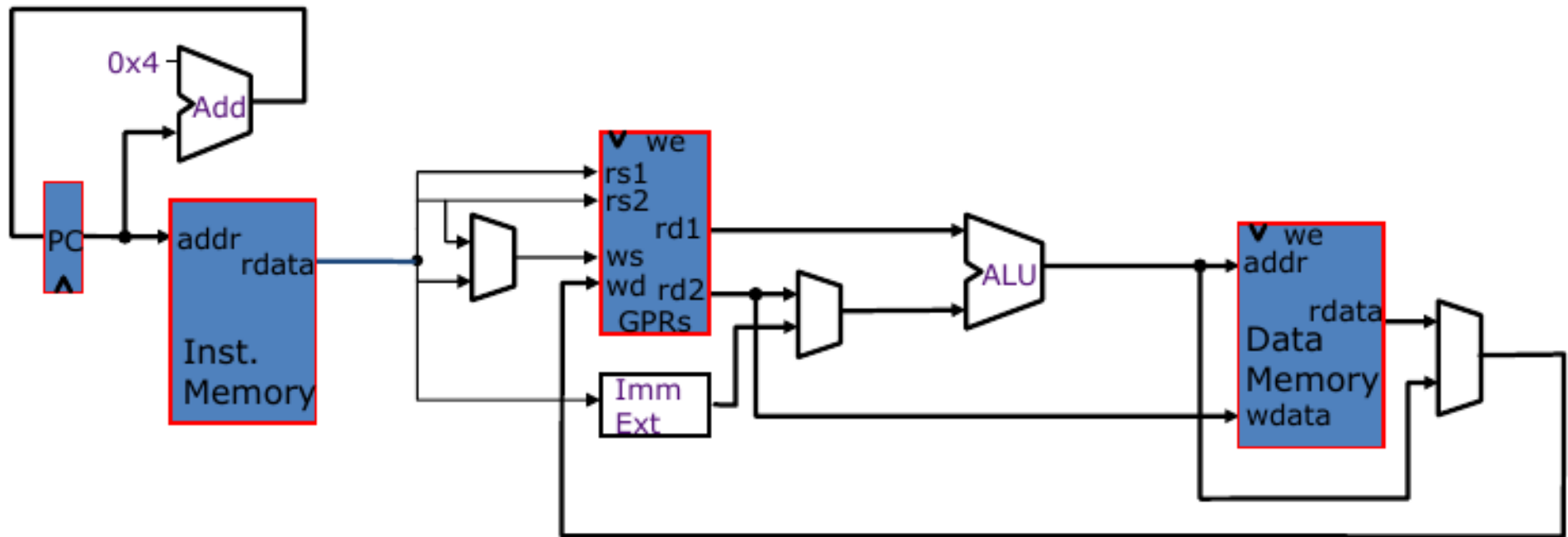


- Todas las instrucciones pasan por los mismos estados.
- No se comparten recursos entre dos etapas.
- Los retardos de propagación es igual en todos los estados.
- La transición de un estado que entra de una instrucción no se afecta por las transiciones de otras etapas.

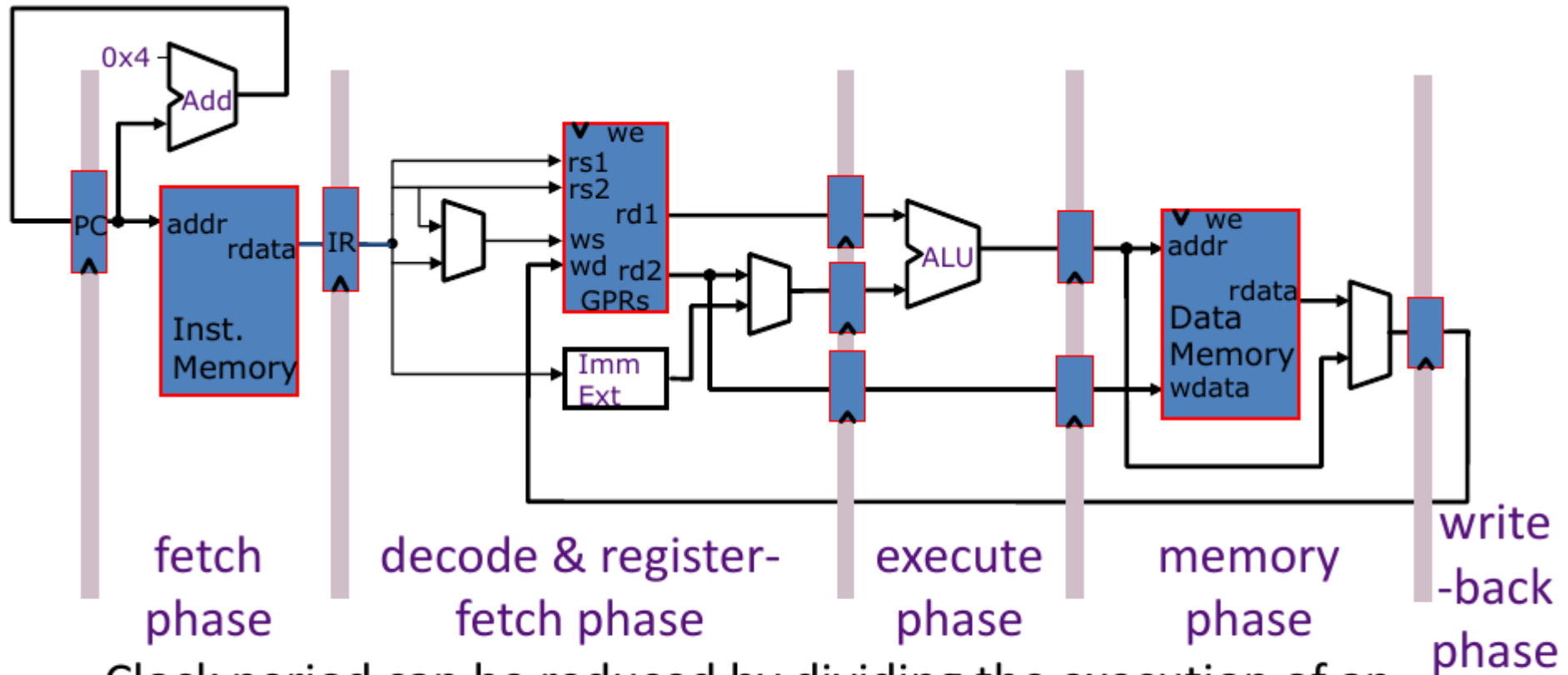
FILOSOFÍA PIPELINE



ESTADO PIPELINE SIMPLIFICADO



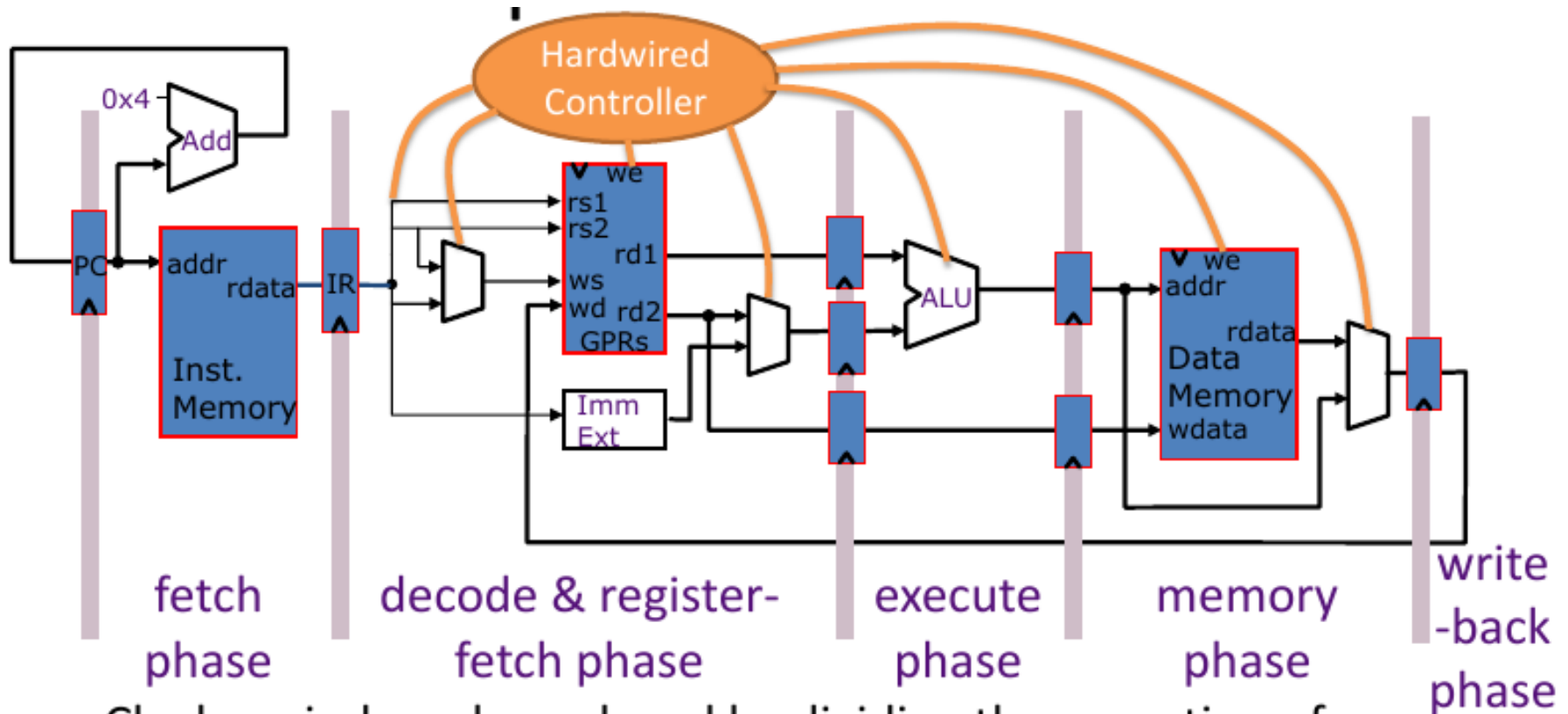
ESTADO PIPELINE



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

$$t_c > \max \{t_{IM}, t_{RF}, t_{ALU}, t_{DM}, t_{RW}\} (= t_{DM} \text{ probably})$$

ESTADO PIPELINE.



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

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

LEY DEL RENDIMIENTO DEL PROCESADOR

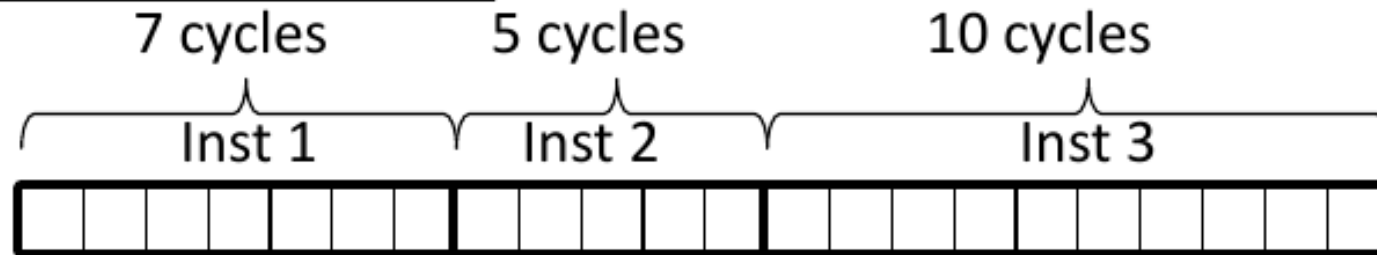
$$\frac{\text{Tiempo}}{\text{Programa}} = \frac{\text{Instrucciones}}{\text{Programa}} * \frac{\text{Ciclo}}{\text{Instrucción}} * \frac{\text{Tiempo}}{\text{Ciclo}}$$

- 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

Microcoded machine



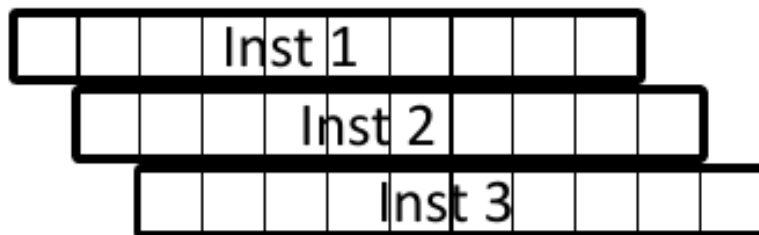
3 instructions, 22 cycles, $CPI=7.33$

Unpipelined machine



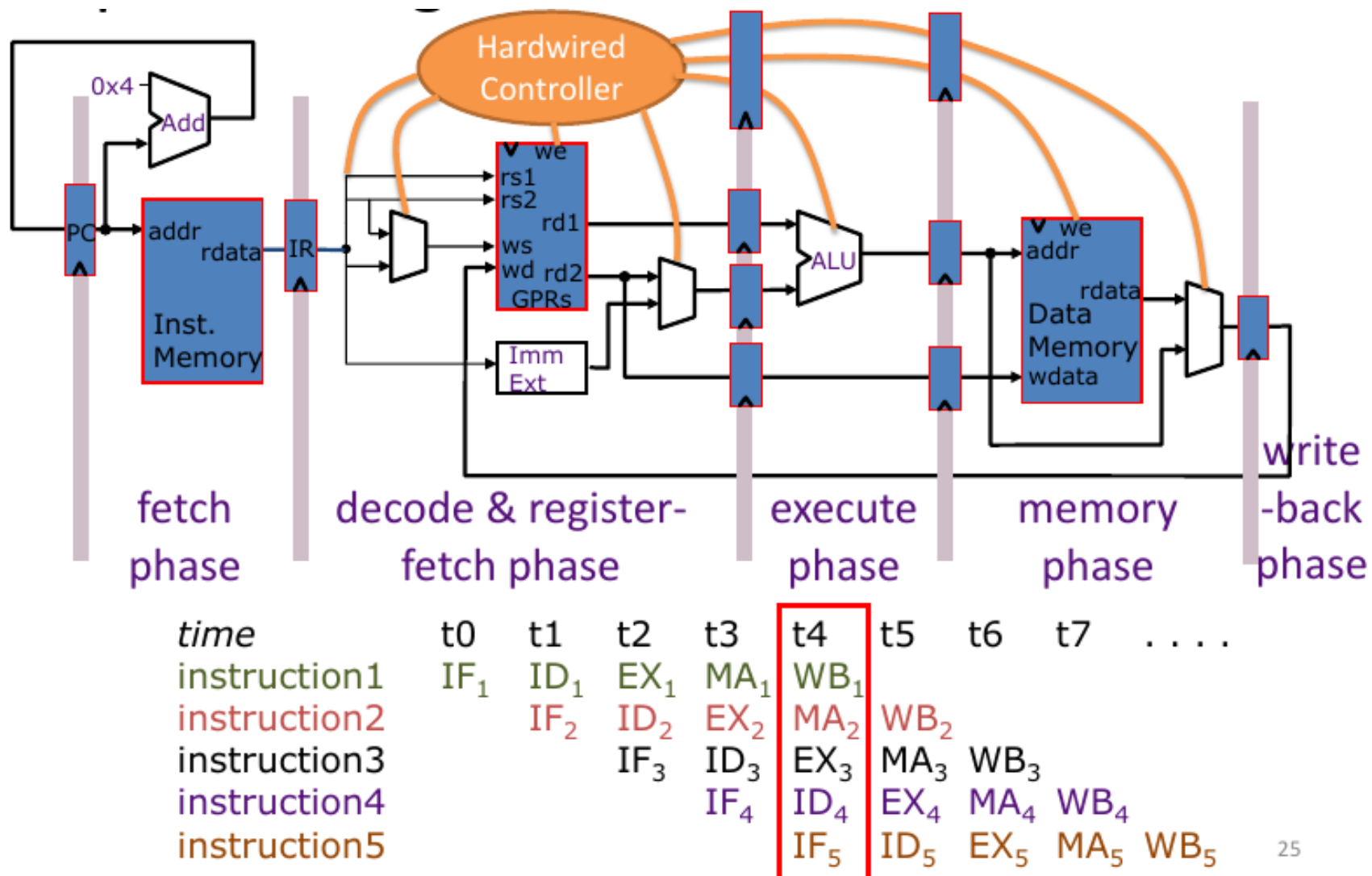
3 instructions, 3 cycles, $CPI=1$

Pipelined machine



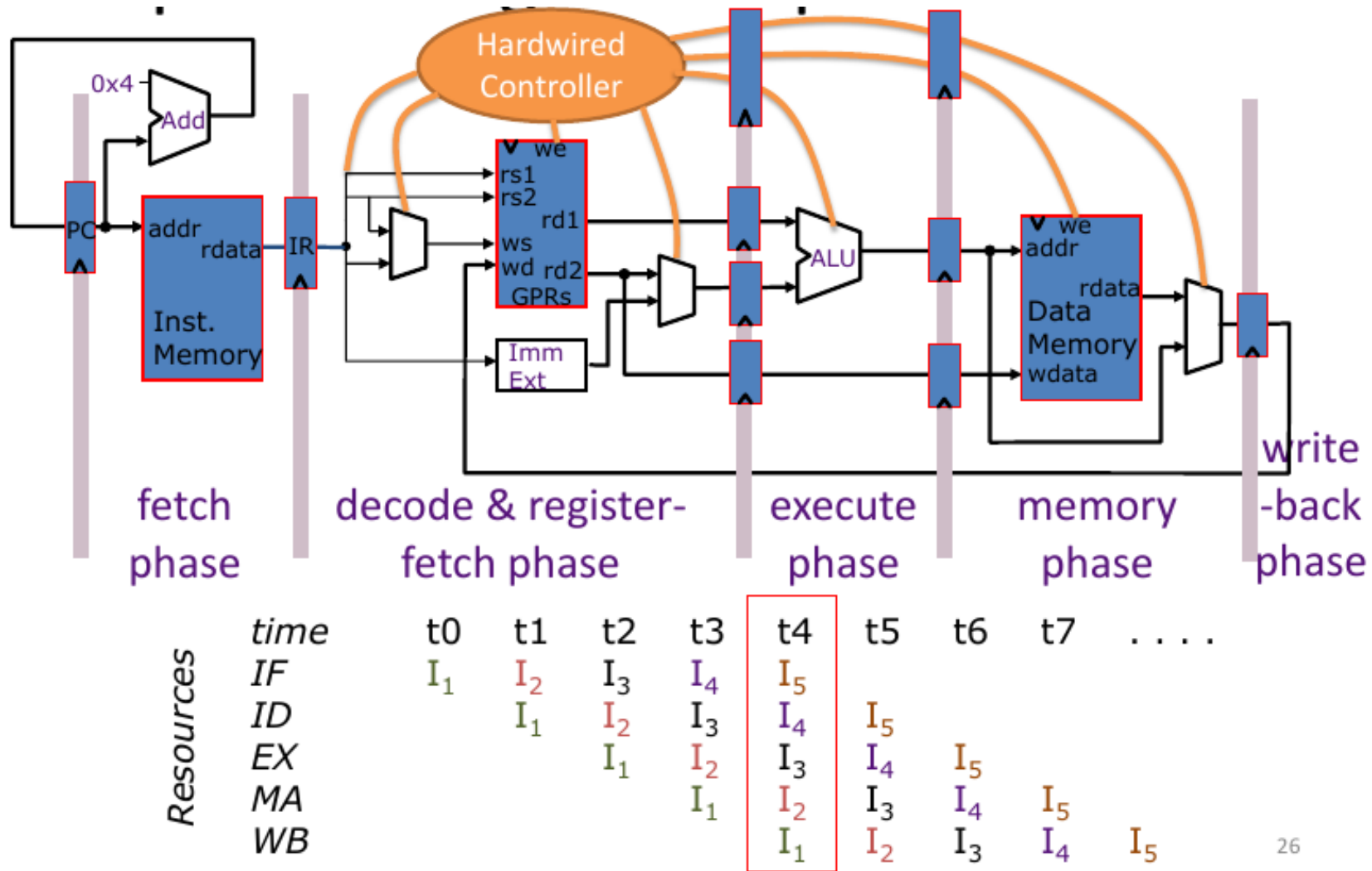
3 instructions, 3 cycles, $CPI=1$

ESTADO PIPELINE: TRANSICIONES Vs TIME.



25

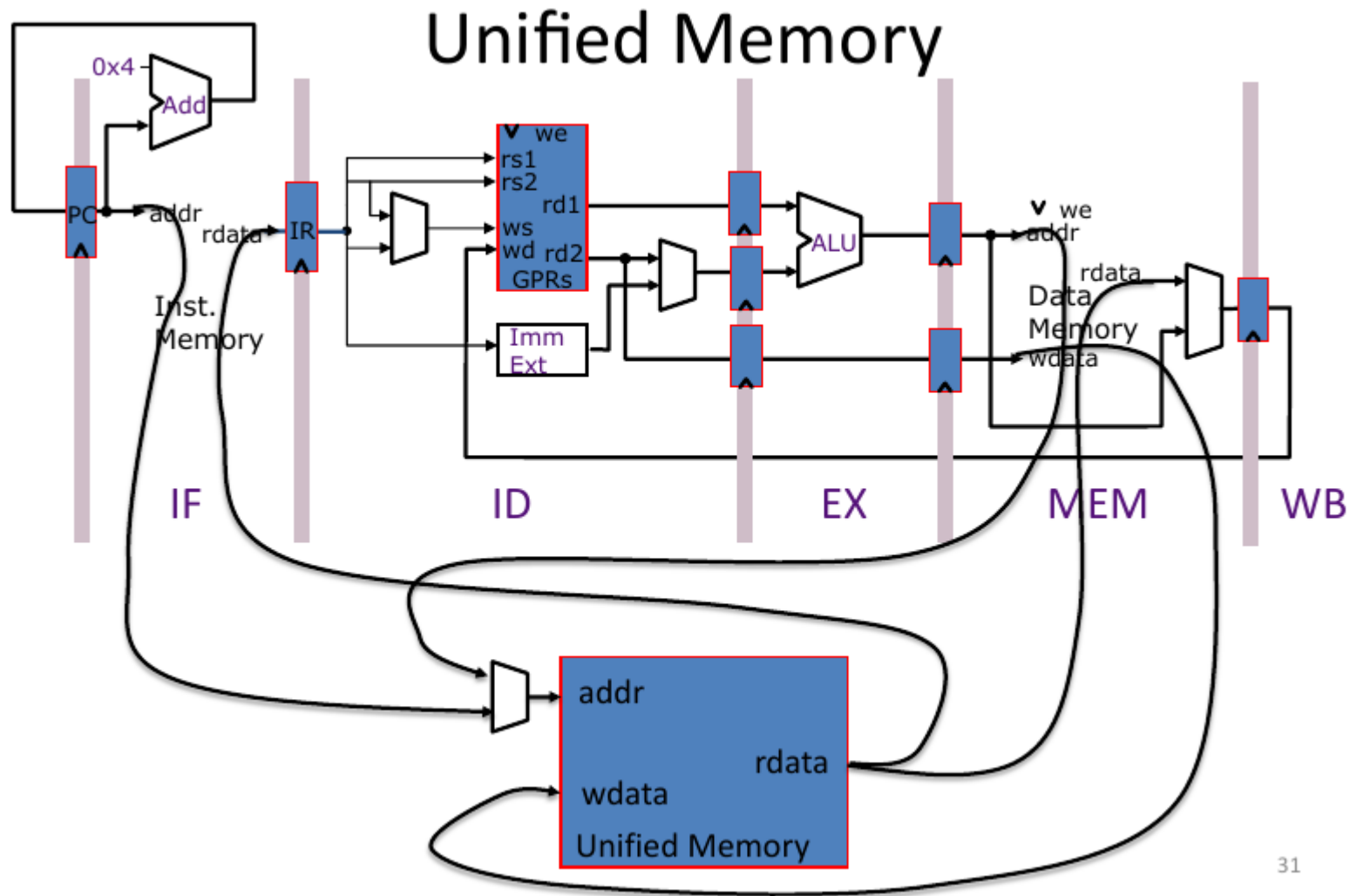
ESTADO PIPELINE: ESTADO Vs TIME.



PROBLEMAS DEL ESTADO PIPELINE.

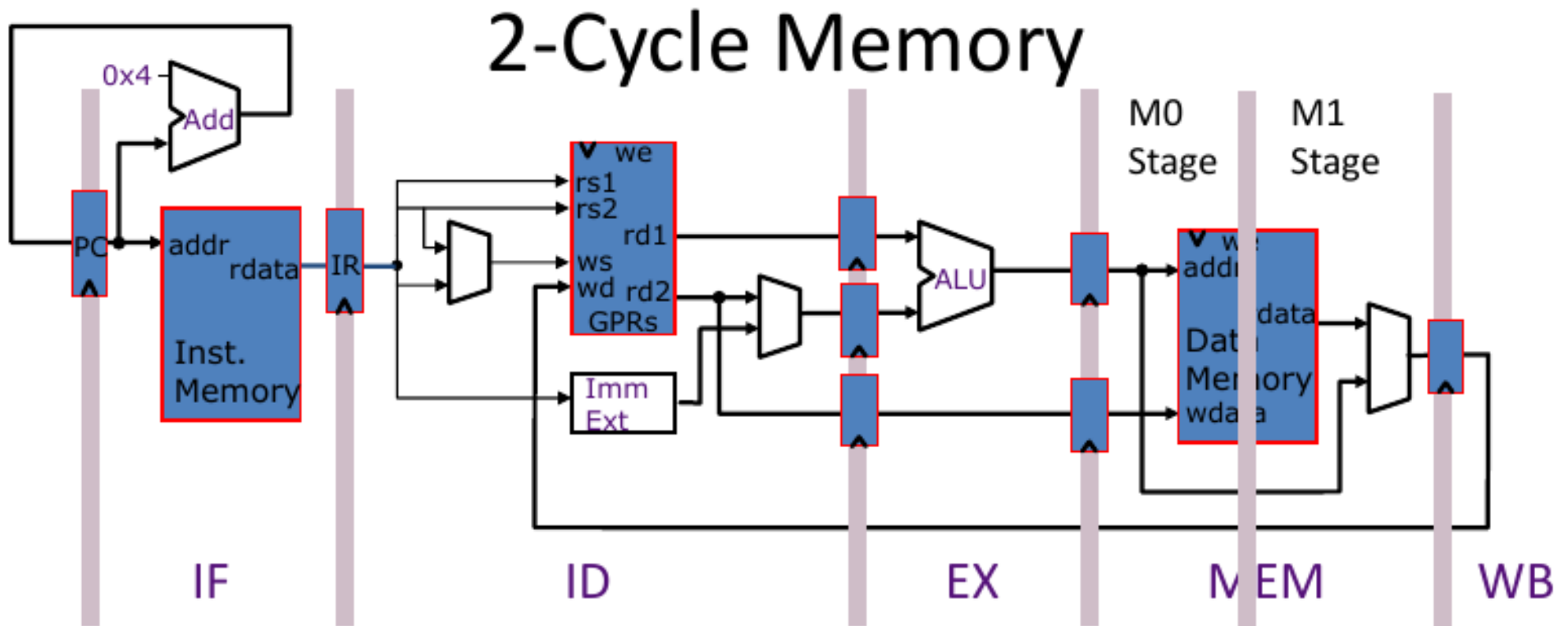
- **ESTRUCTURALES:** DOS INSTRUCCIONES DIFERENTES RECURREN A LOS MISMOS RECURSOS DE HARDWARE EN LOS MISMOS ESTADO DE TIEMPO.
- **DATOS:** UNA INSTRUCCIÓN DEPENDE DE DATOS GENERADOS EN LA EJECUCIÓN DE LA INSTRUCCIÓN ANTERIOR DEL PIPELINE.
- **CONTROL:** EL ESTADO PIPELINE SE INTERRUMPE POR LA NO SECUENCIA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

PROBLEMAS DEL ESTADO PIPELINE: ESTRUCTURAL

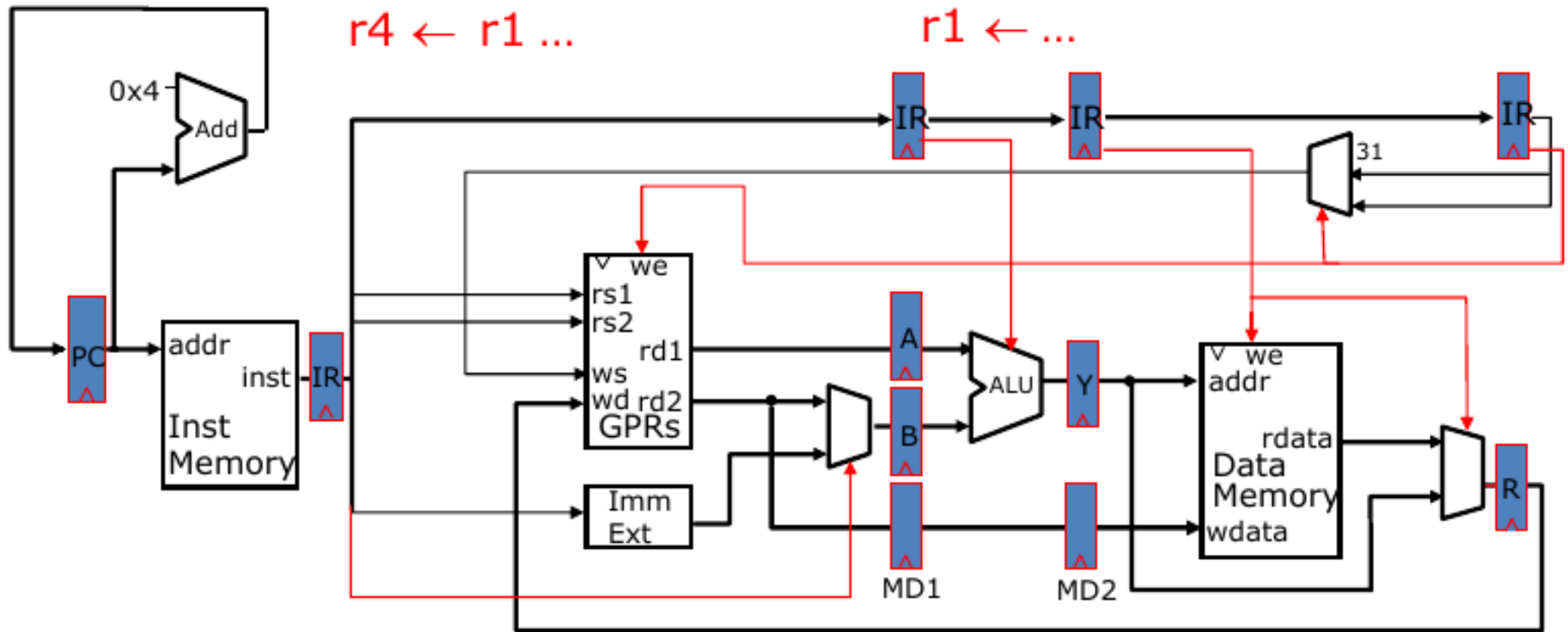


31

PROBLEMAS DEL ESTADO PIPELINE: ESTRUCTURAL



PROBLEMAS DEL ESTADO PIPELINE: DATOS



...

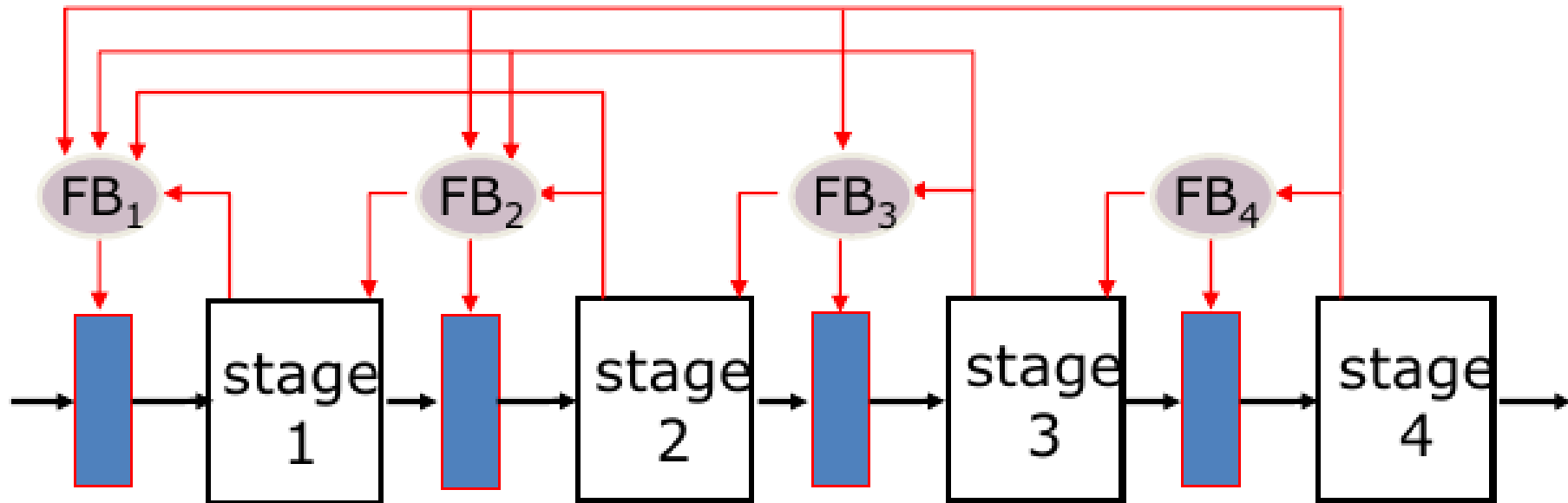
$r1 \leftarrow r0 + 10$ (ADDI R1, R0, #10)

$r4 \leftarrow r1 + 17$ (ADDI R4, R1, #17)

...

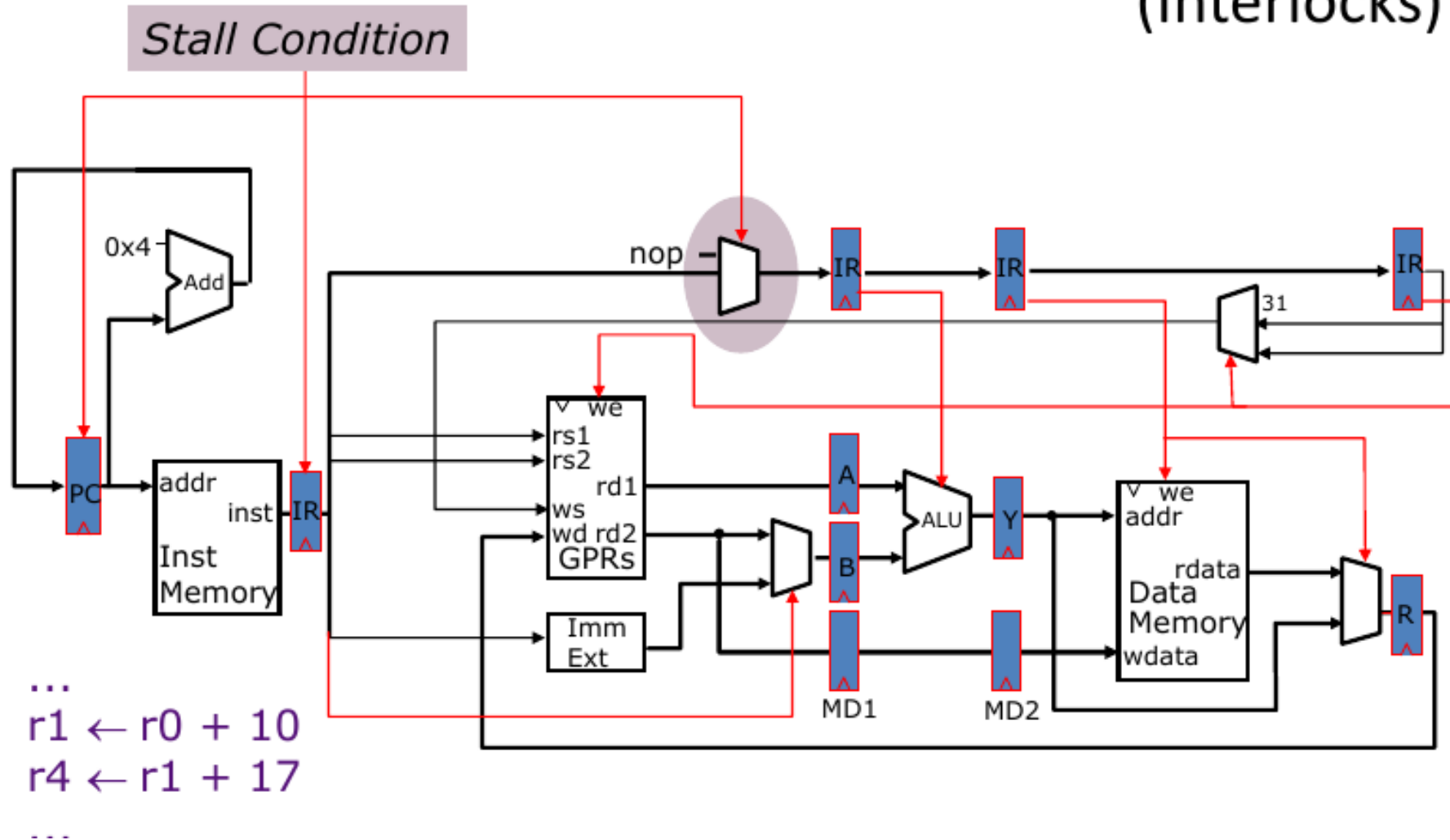
$r1$ is stale. Oops!

PROBLEMAS DEL ESTADO PIPELINE: RETRO ALIMENTACIÓN

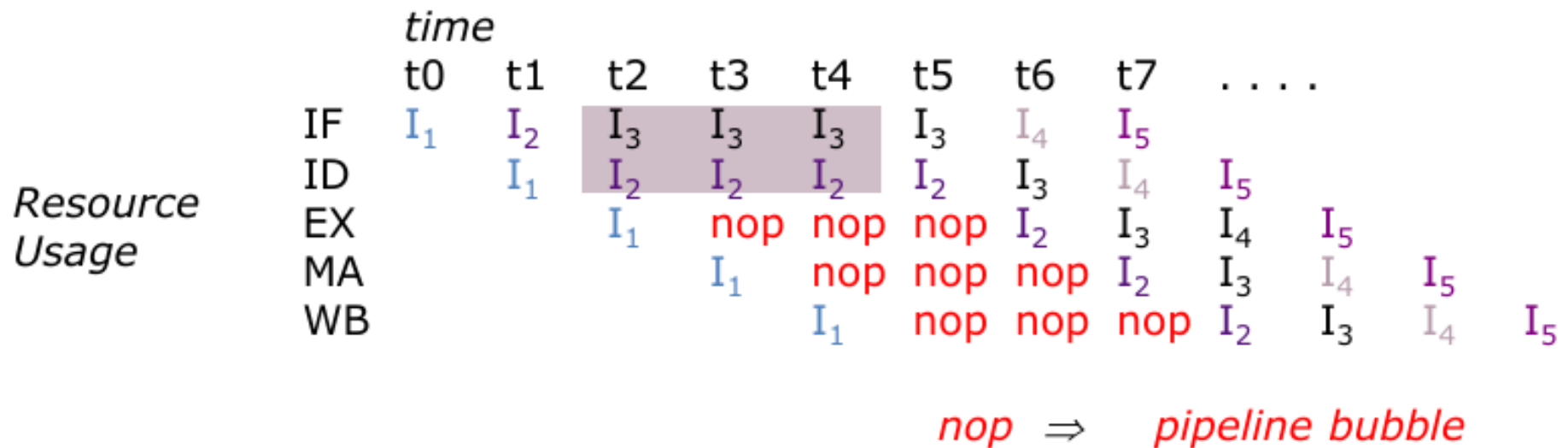
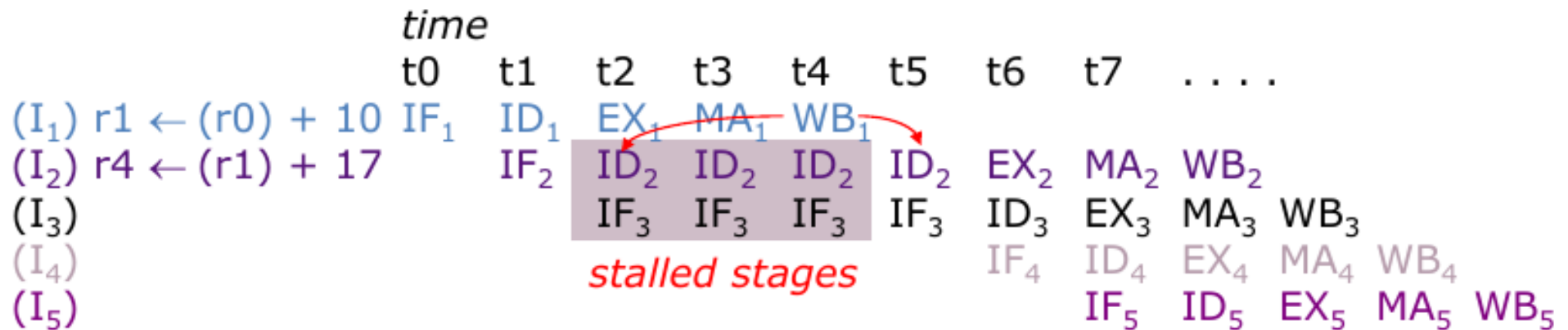


PROBLEMA DE ESTADO PIPELINE: RETRO ALIMENTACIÓN

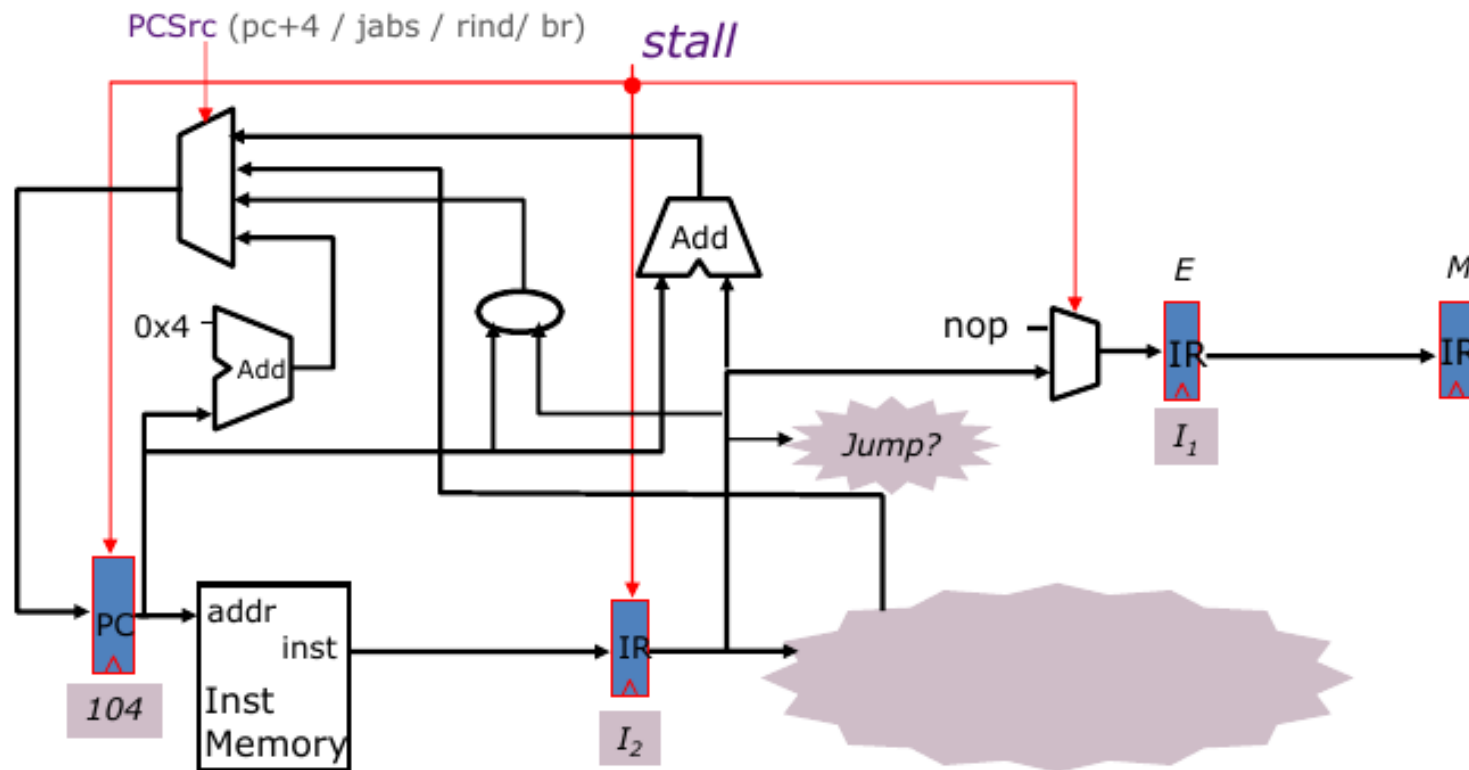
(Interlocks)



PROBLEMAS DE ESTADO PIPELINE: RETRO ALIMENTACIÓN



PROBLEMA ESTADO PIPELINE: CONTROL

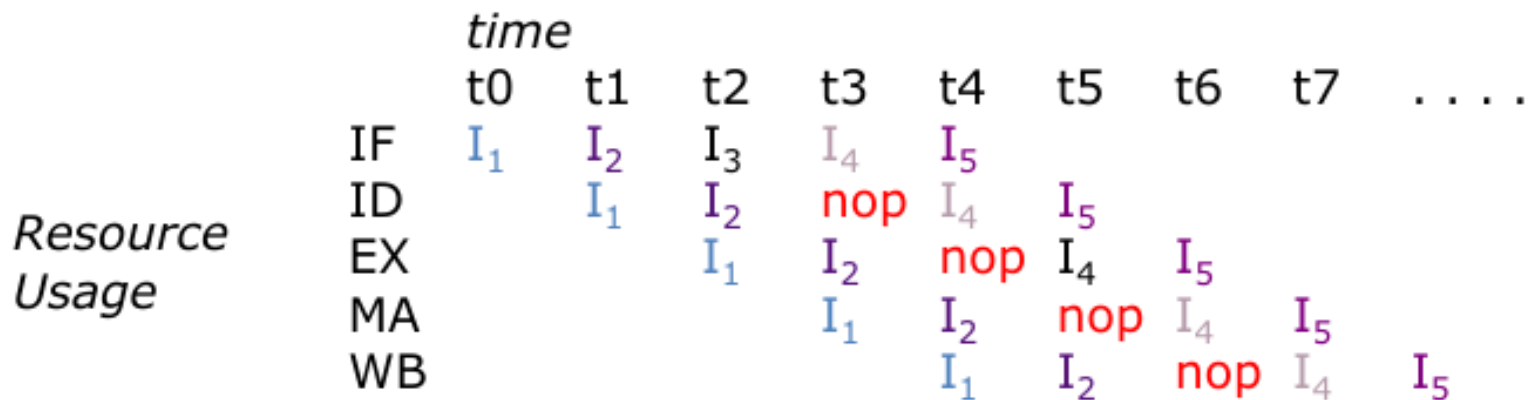
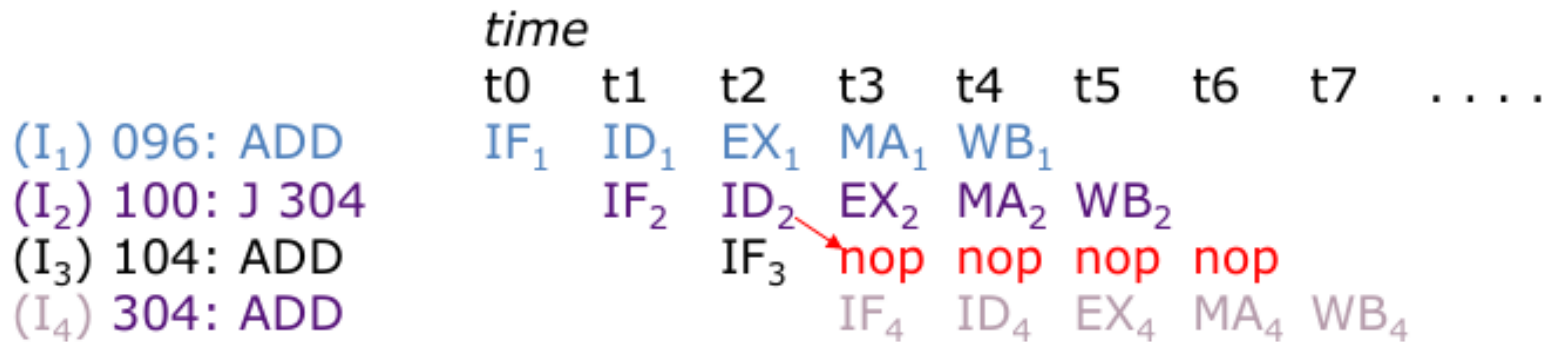


I_1	096	ADD	
I_2	100	J 304	
I_3	104	ADD	<i>kill</i>
I_4	304	ADD	

A jump instruction kills (not stalls)
the following instruction

How?

PROBLEMAS ESTADO PIPELINE: CONTROL



nop ⇒ *pipeline bubble*

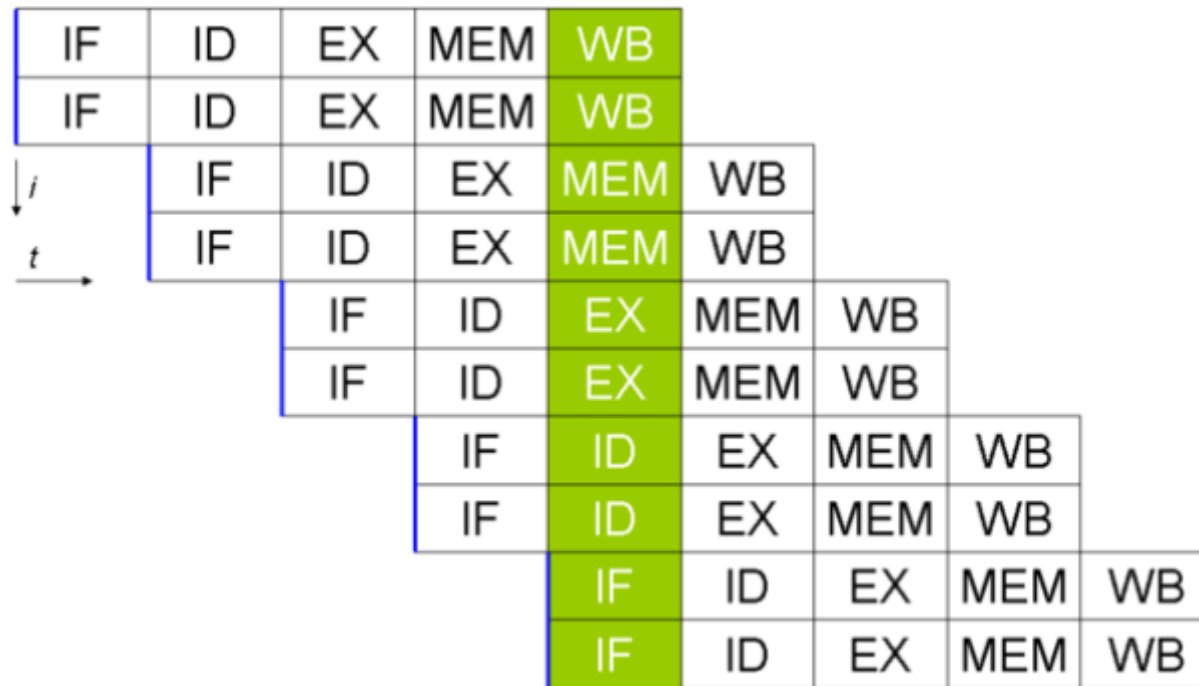
SUPER PIPELINE

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.

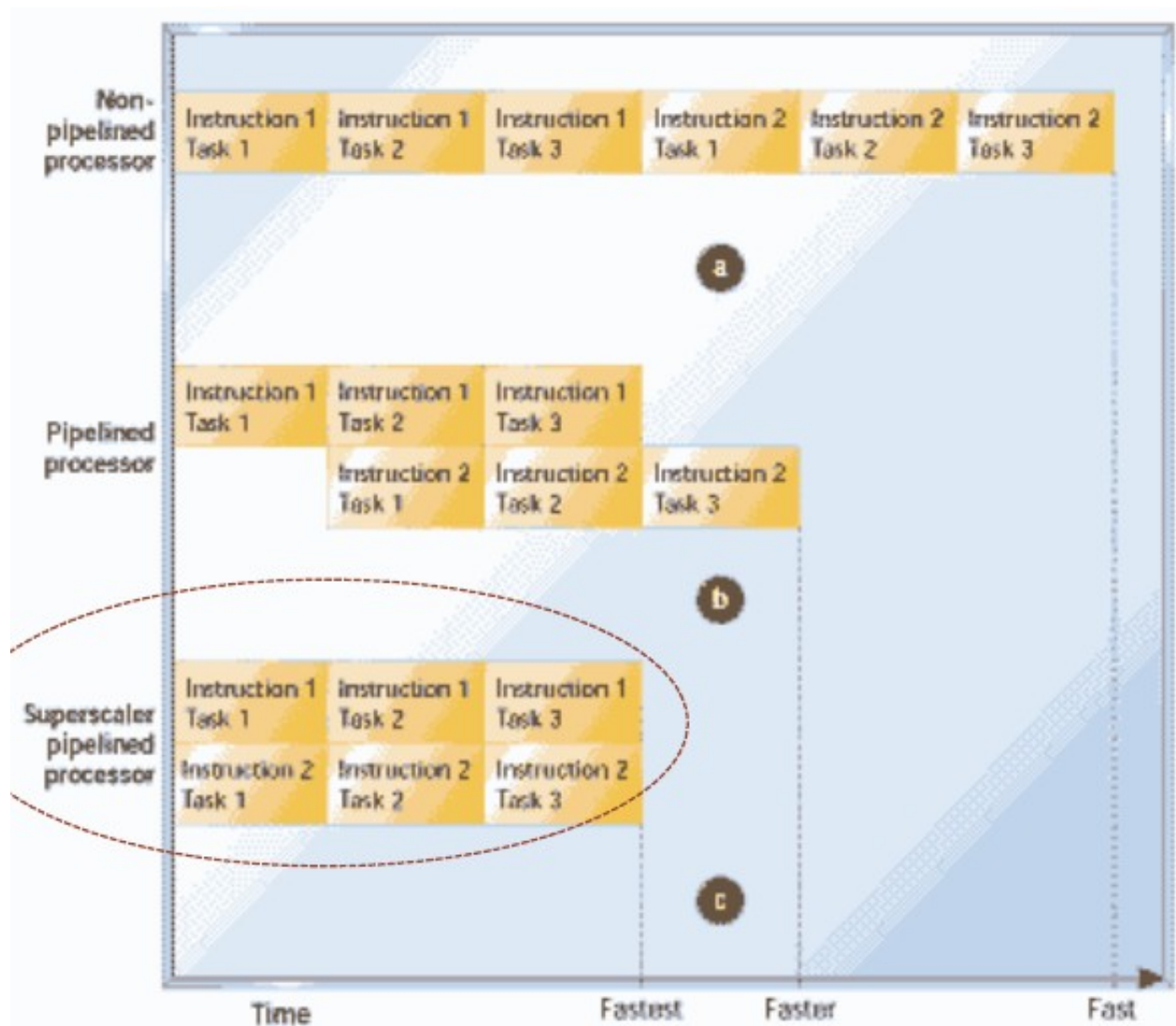
Si la ALU toma tres veces más tiempo que cualquier otro módulo, podemos dividir la ALU en tres etapas separadas, lo que reducirá la cantidad de tiempo perdido en etapas más cortas. El problema aquí es que necesitamos encontrar una manera de subdividir nuestras etapas en etapas más cortas, y también necesitamos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

SUPERESCALER

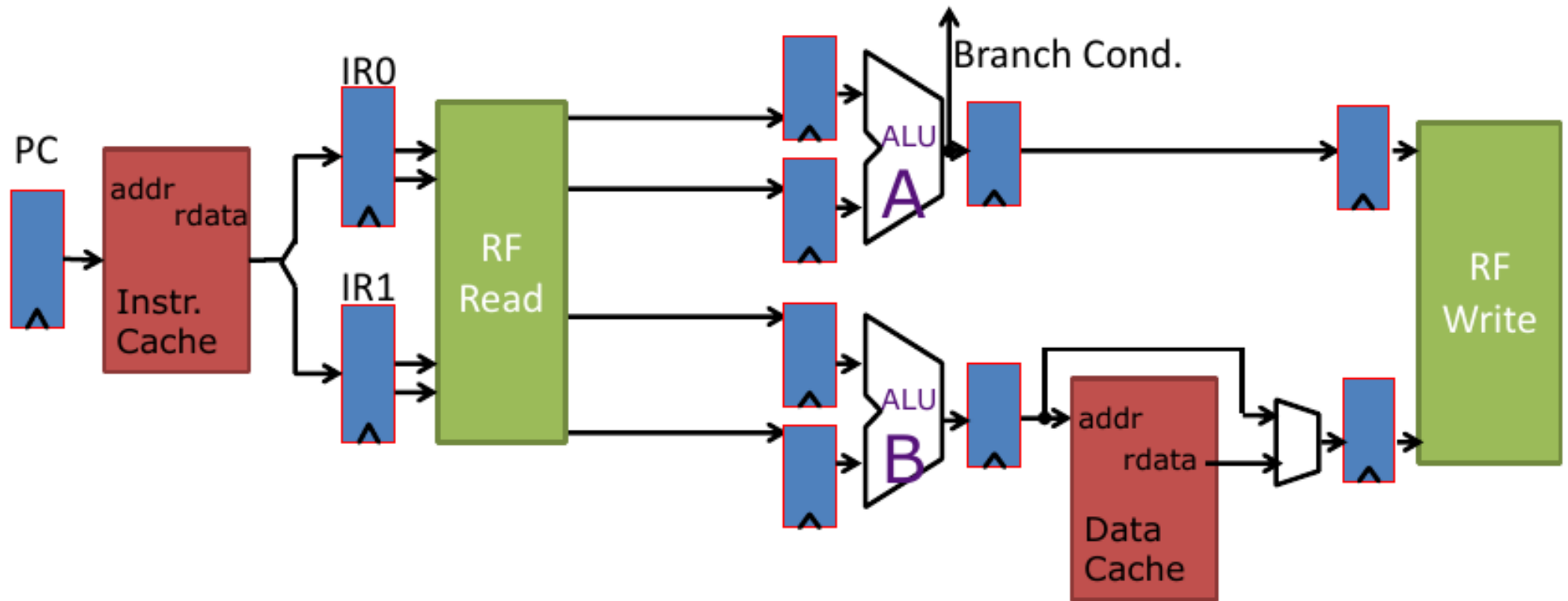
Arquitectura en la que un procesador es capaz de ejecutar mas de una instrucción por ciclo de reloj. En esta arquitectura se replica las unidades para lograr ejecutar múltiples instrucciones.



SUPERESCALER



SUPERESCALAR



SUPERESCALAR

