Electronica digital 2

Arquitectura pipeline procesador

Ferney Alberto Beltrán Molina

Agosto 2020

Contacto

Nombre:	Ferney Alberto Beltrán Molina, Ing, MSc, PhD(c)
Email:	fabeltranm@unal.edu.co
oficina:	

Contenido

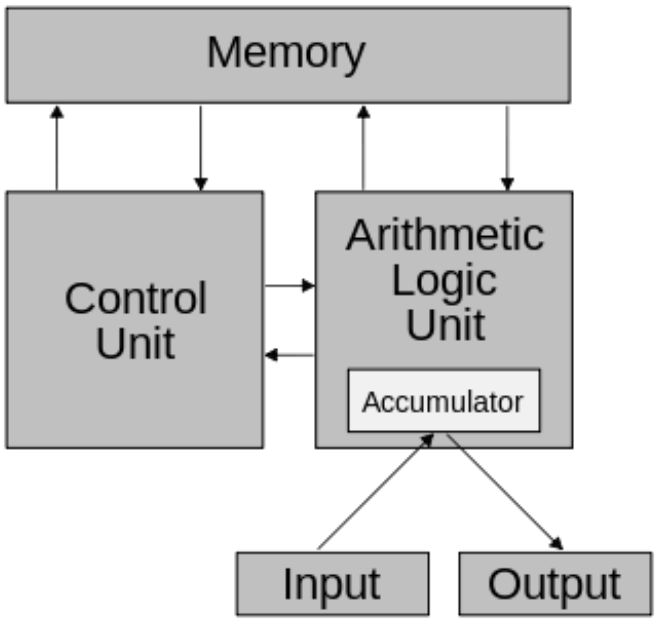
Índice

Antes..

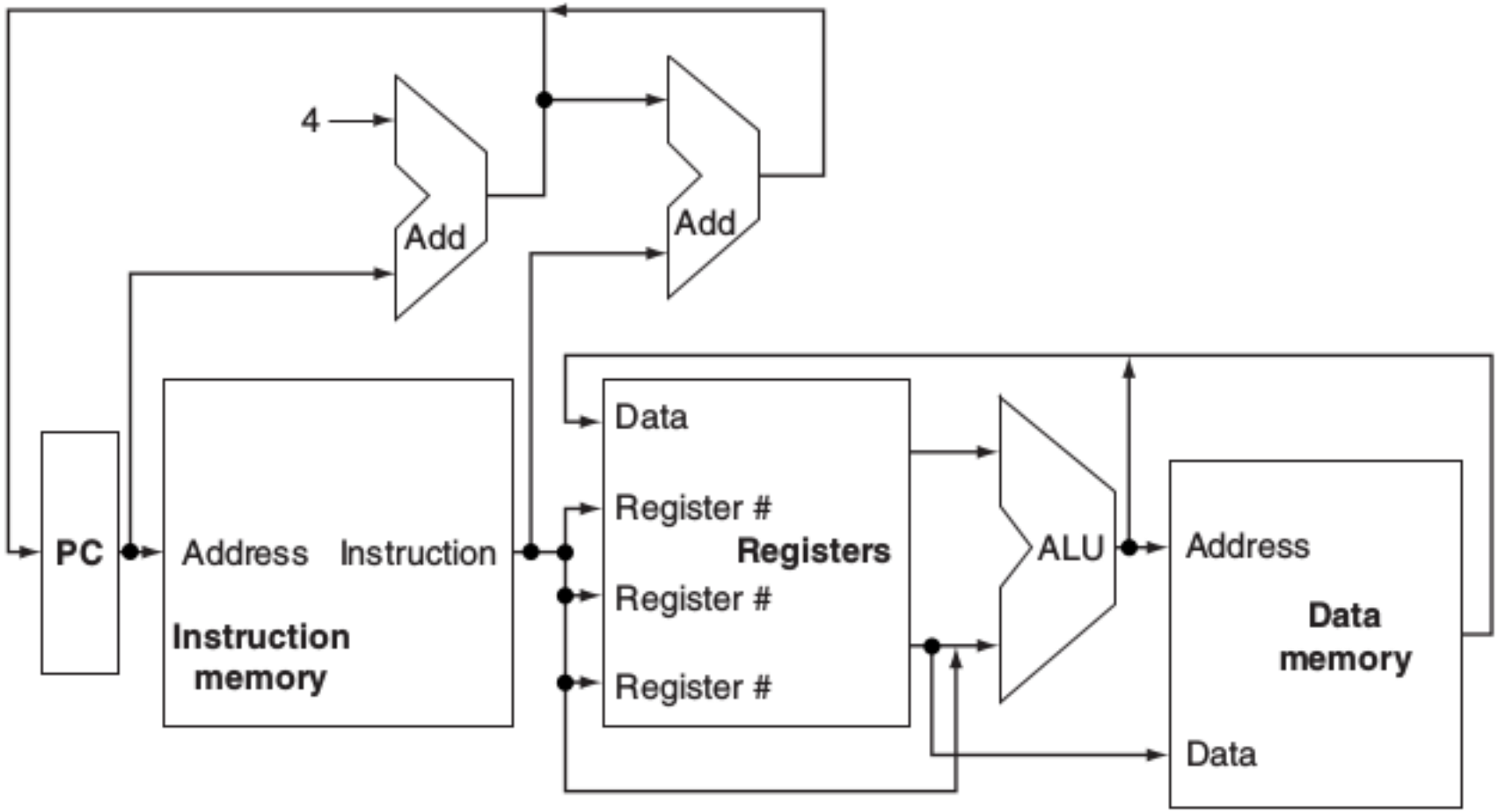
En la clase anterior vimos:

- Se hablar de la frontera Hw/Sw
- Se presenta la arquitectura básica del procesador
- Se presenta los componentes del procesador
- Introducción a la conexión SoC

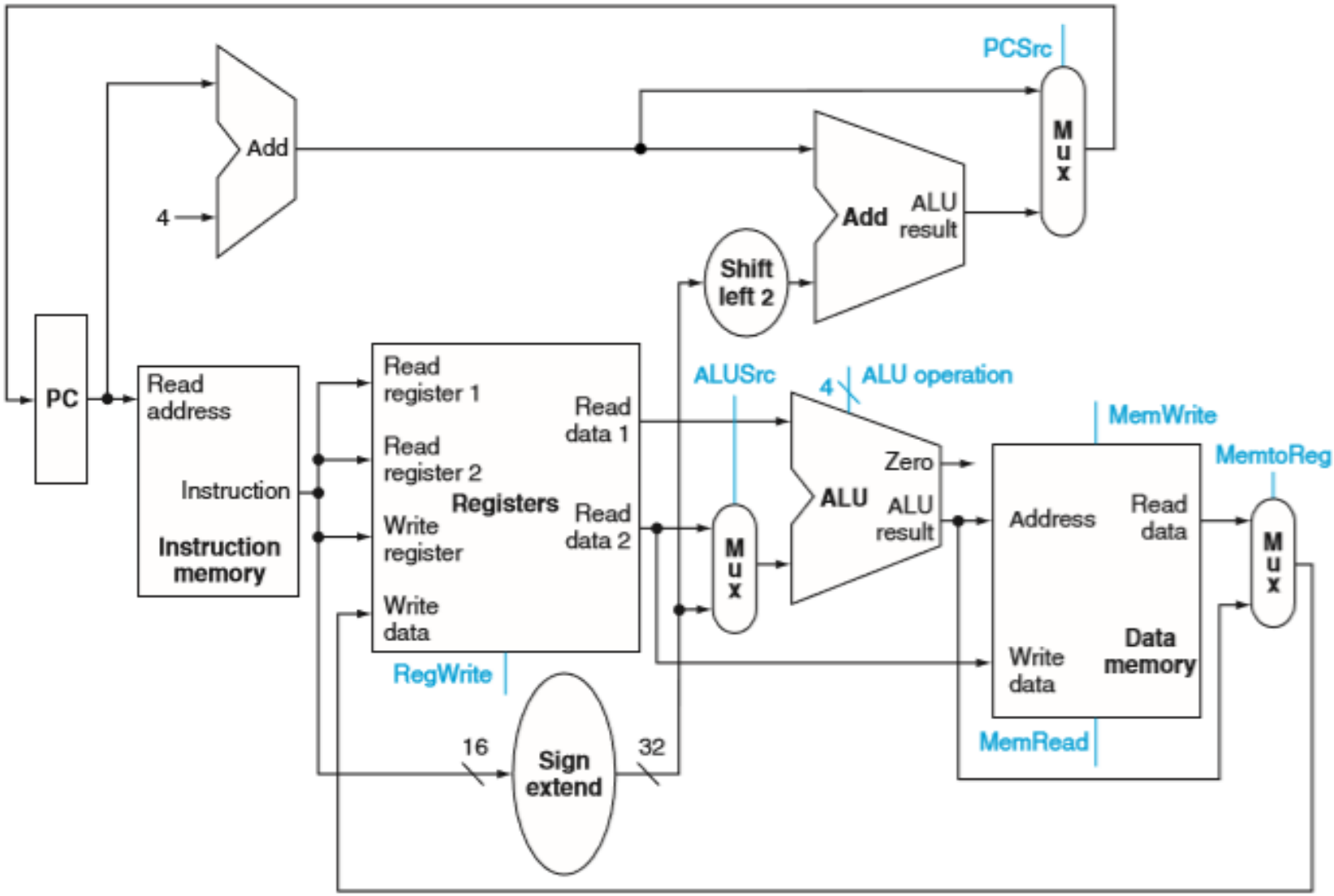
Hardware Software Interface



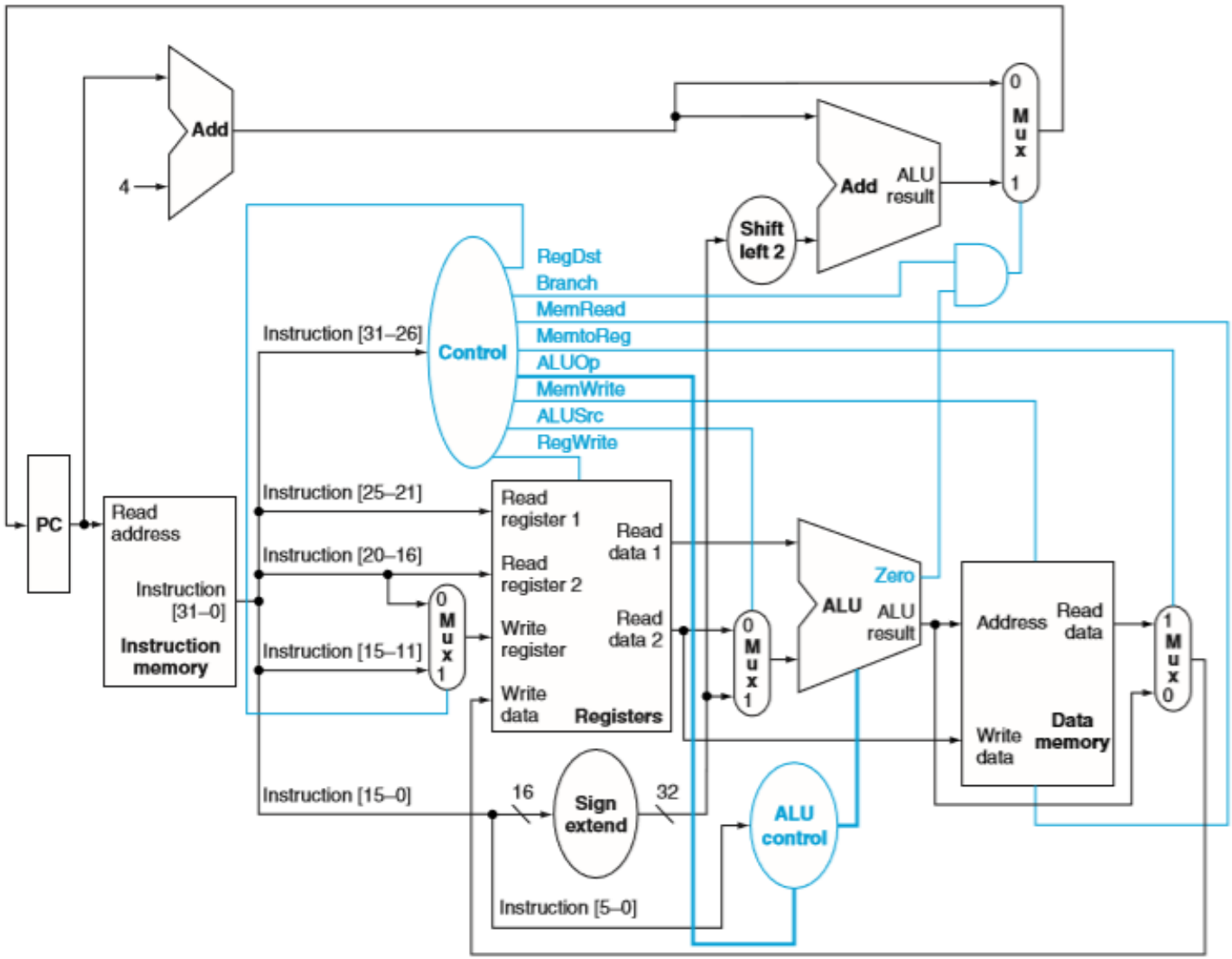
Datapath básico



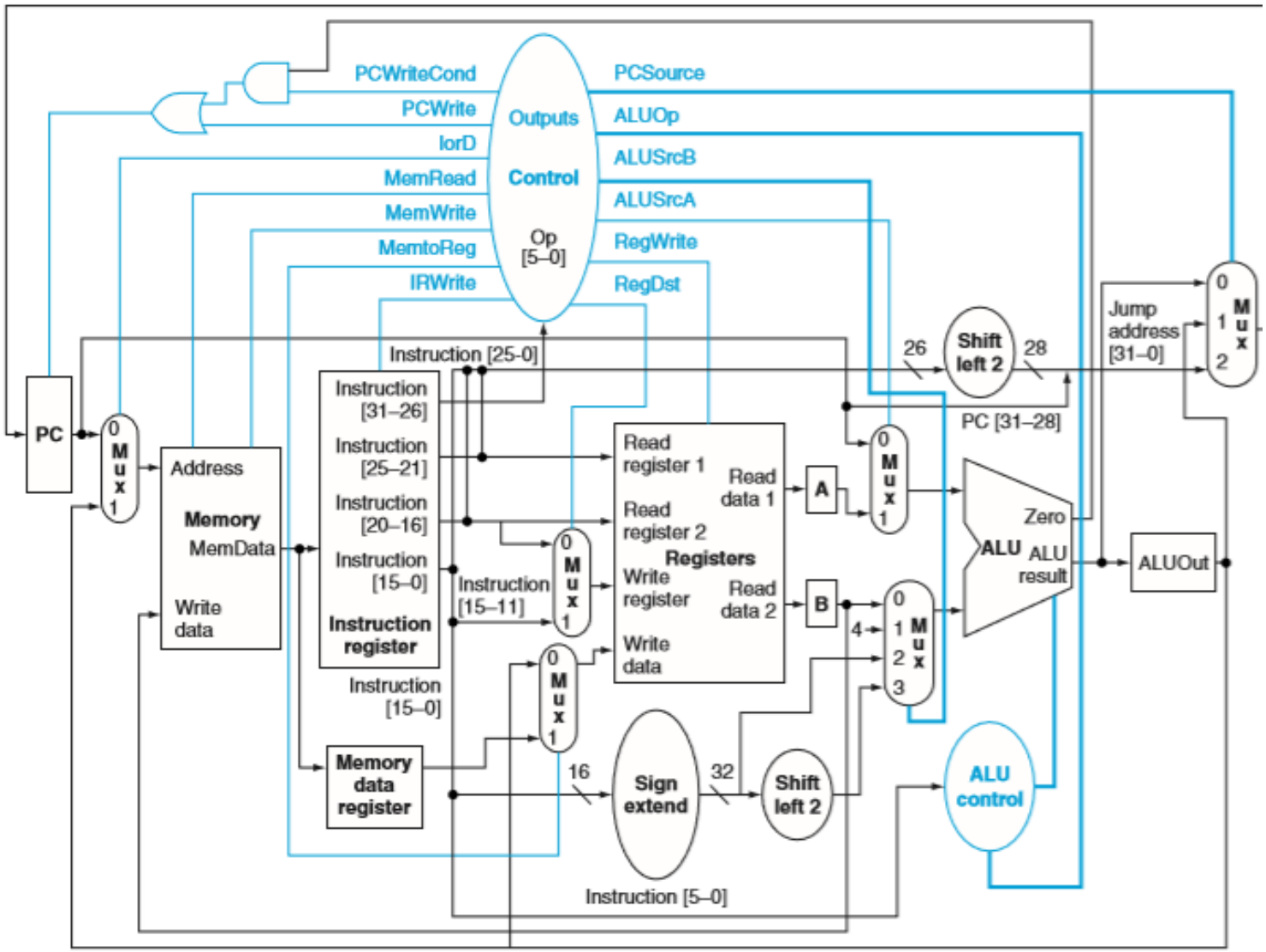
Unidad de control



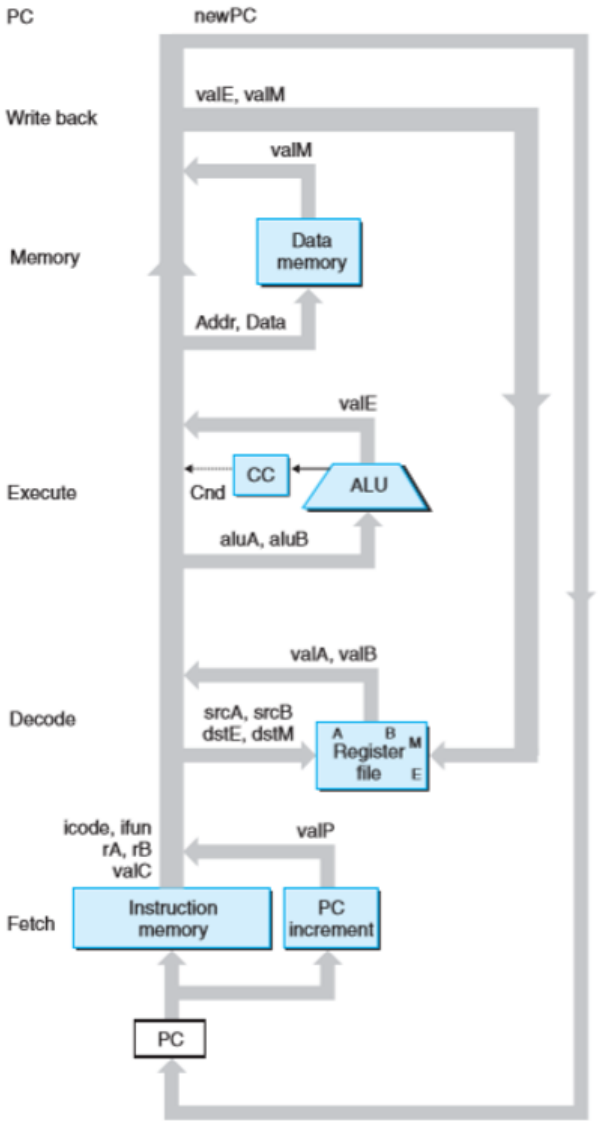
Procesador

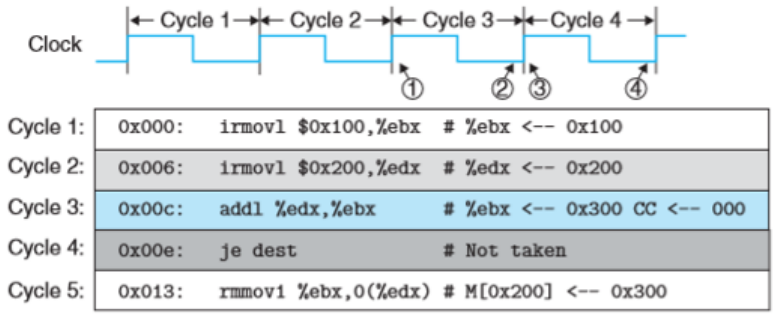

Add
4
Add
ALU result
Shift left 2
M u x
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory

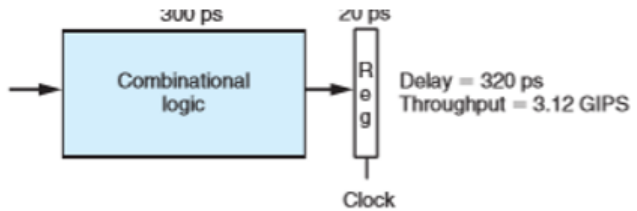
Procesador Multiciclo (Fech Decode Execute)



secuencia (Fech Decode Execute)

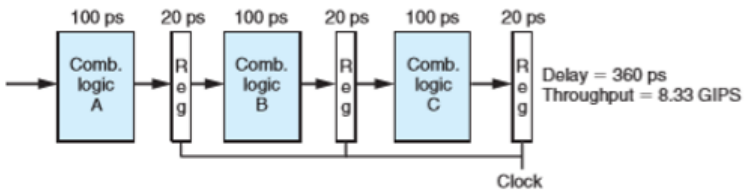






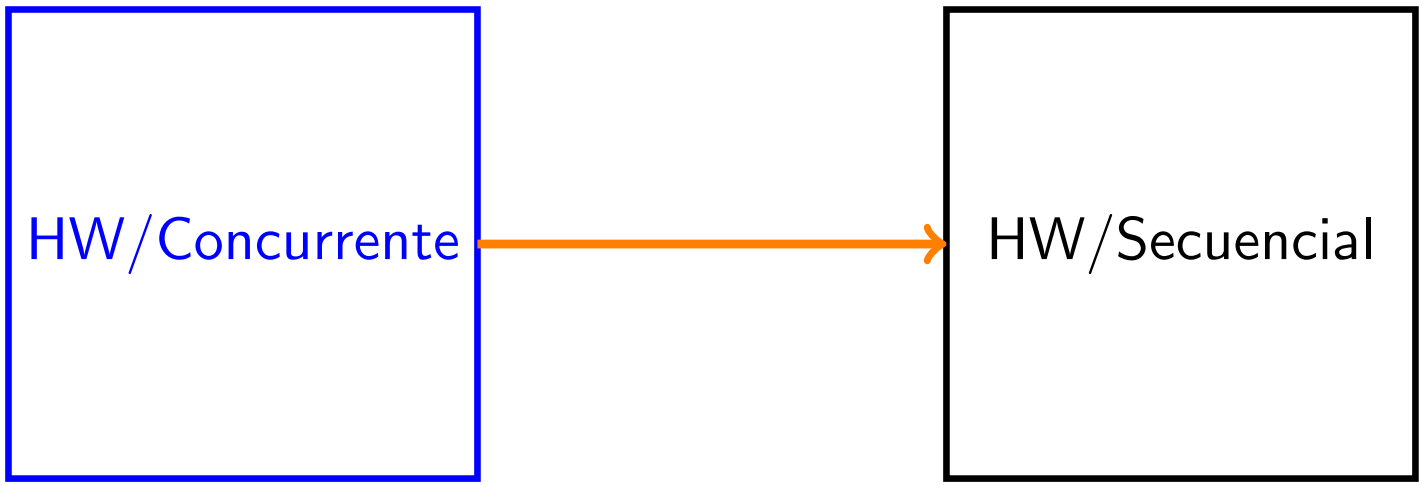
(a) Hardware: Unpipelined

(b) Pipeline diagram



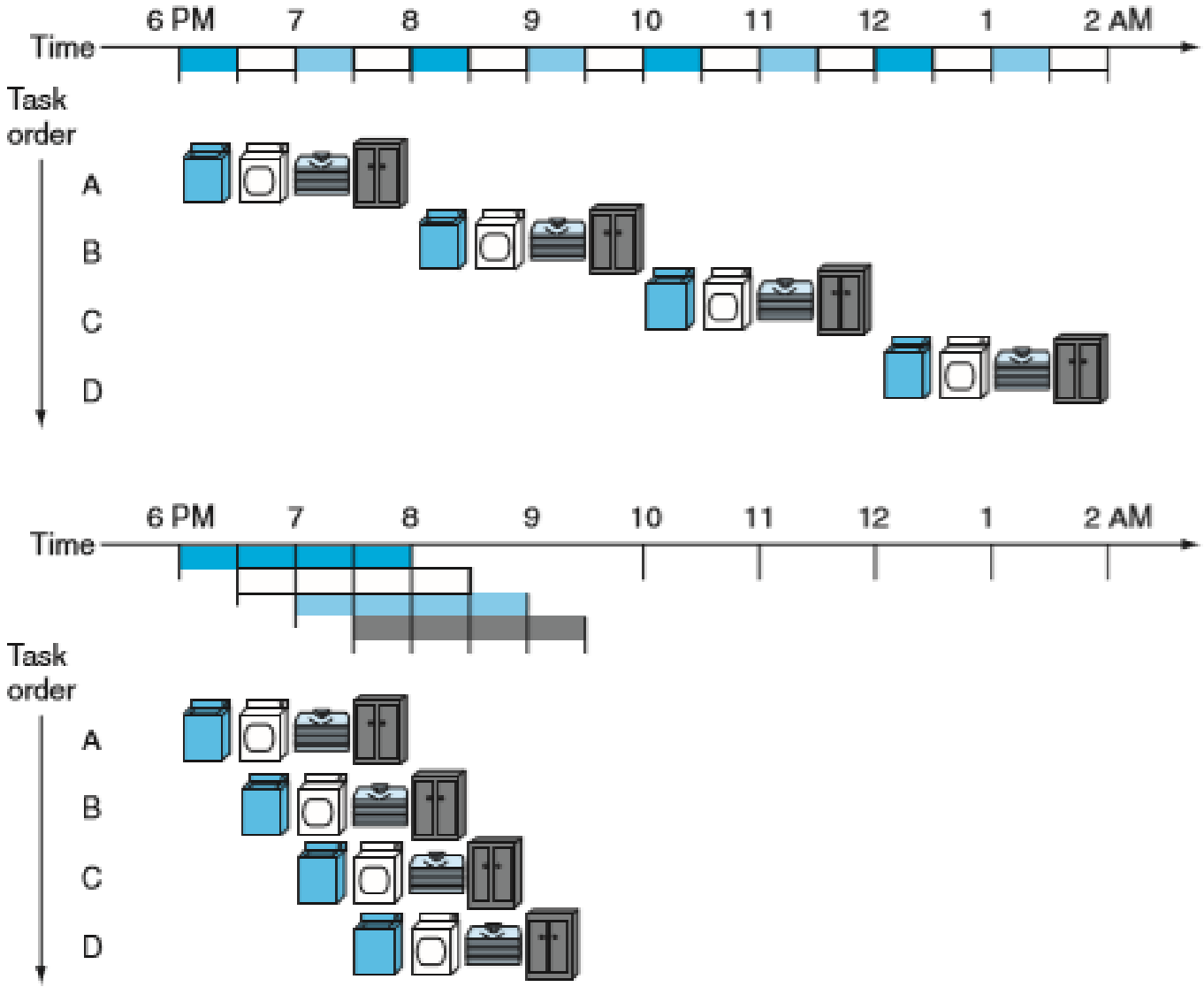
Índice

Concurrente a Secuencial

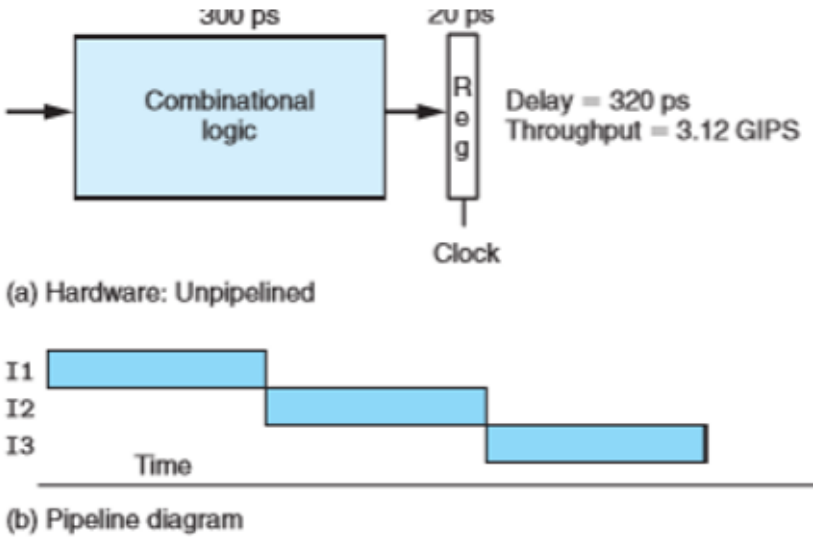


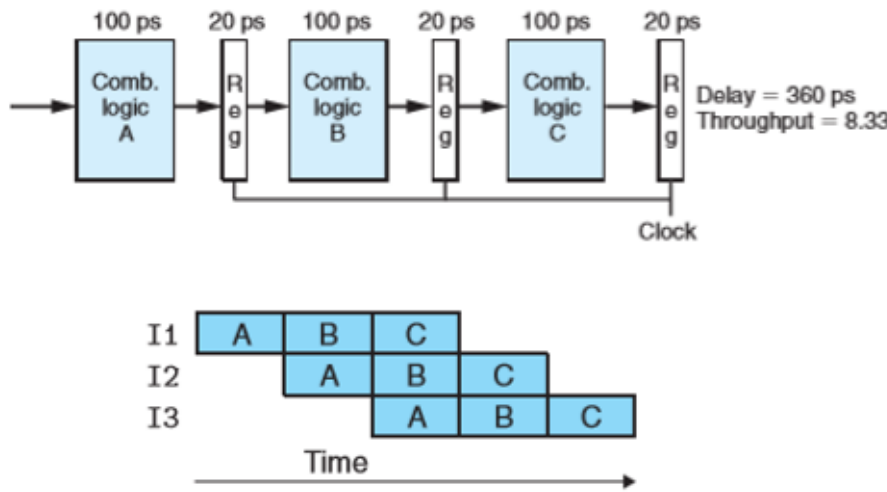
Solución:
Dividir el proceso en etapas independientes
Mover objetos a través de etapas en secuencia
En cualquier momento dado, se procesan varios objetos

concepto

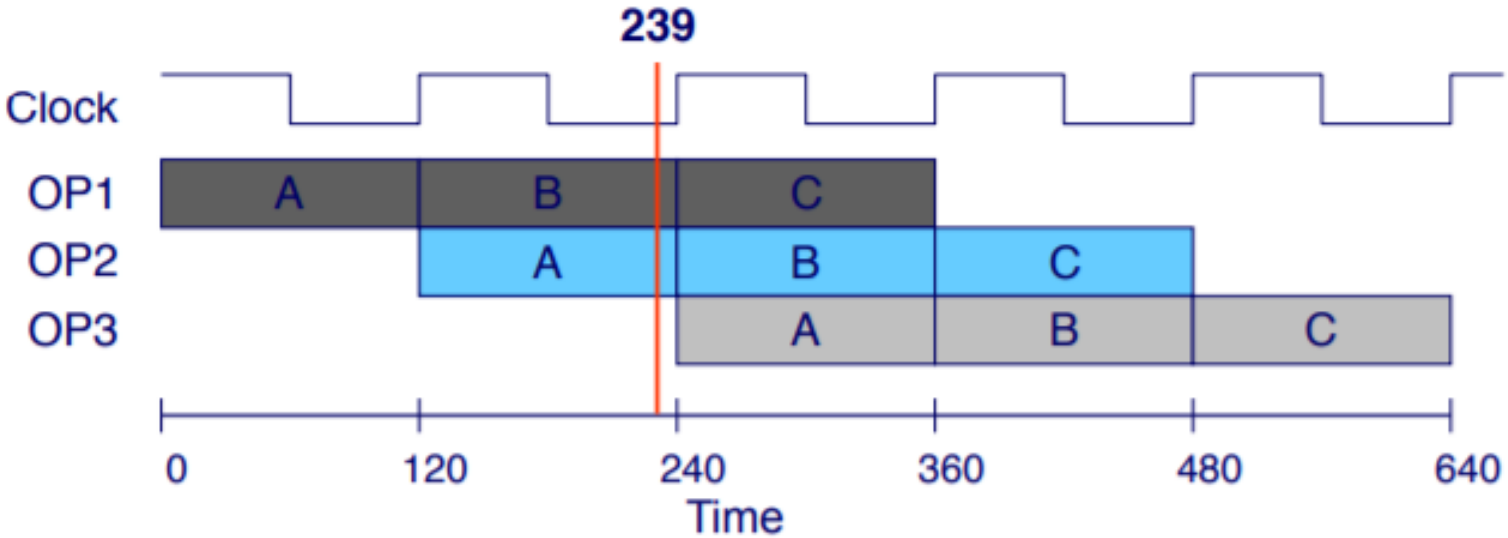

6 PM
7
8
9
10
11
12
1
2 AM
Time
Task order
A
B
C
D
6 PM
7
8
9
10
11
12
1
2 AM
Time
Task order
A
B
C
D

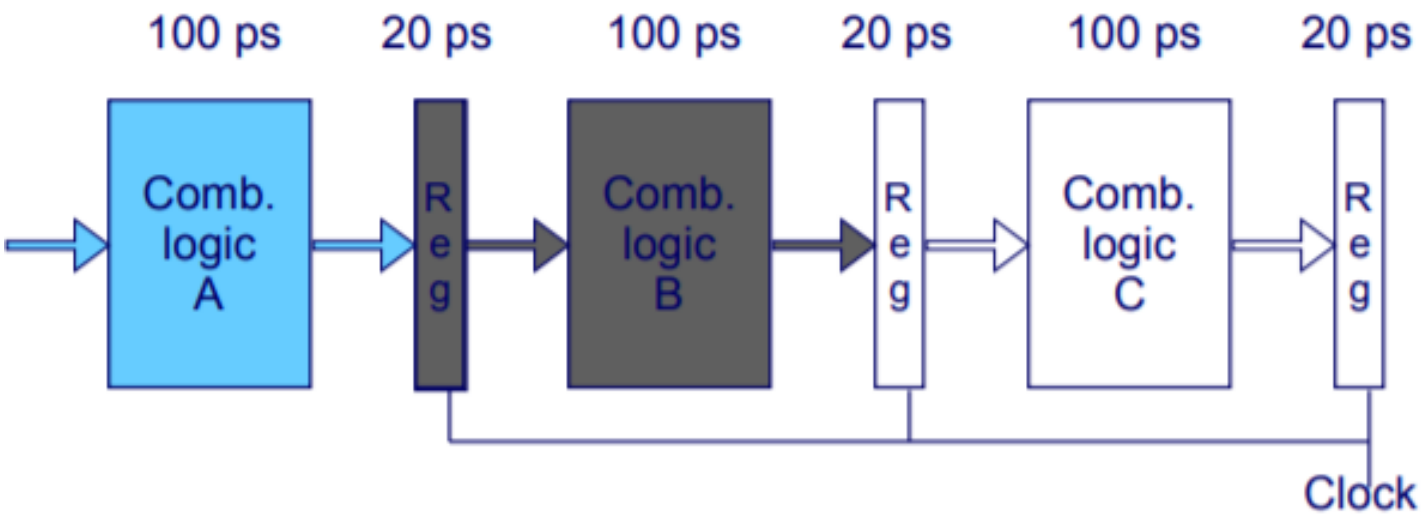
Solución


300 ps
20 ps
Combinational logic
Reg
Delay = 320 ps
Throughput = 3.12 GIPS
Clock
(a) Hardware: Unpipelined
I1
I2
I3
Time
(b) Pipeline diagram

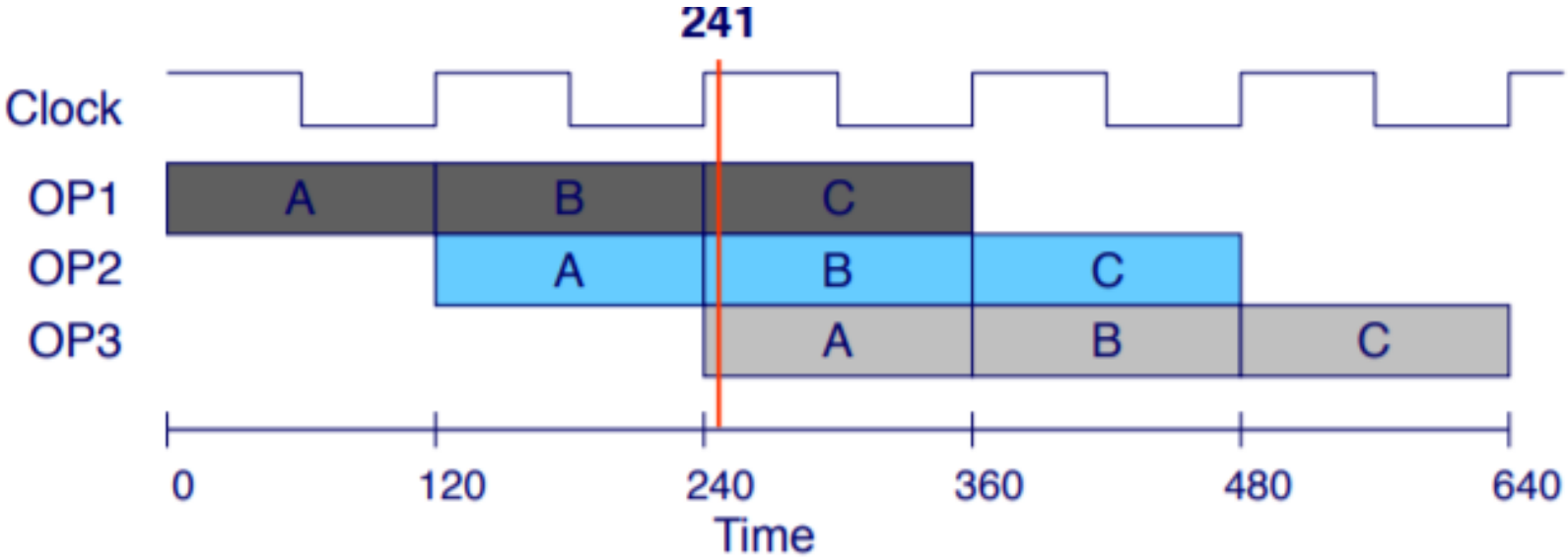

100 ps
20 ps
100 ps
20 ps
100 ps
20 ps
Comb. logic A
Reg
Comb. logic B
Reg
Comb. logic C
Reg
Delay = 360 ps
Throughput = 8.33
Clock
I1
I2
I3
A
B
C
Time

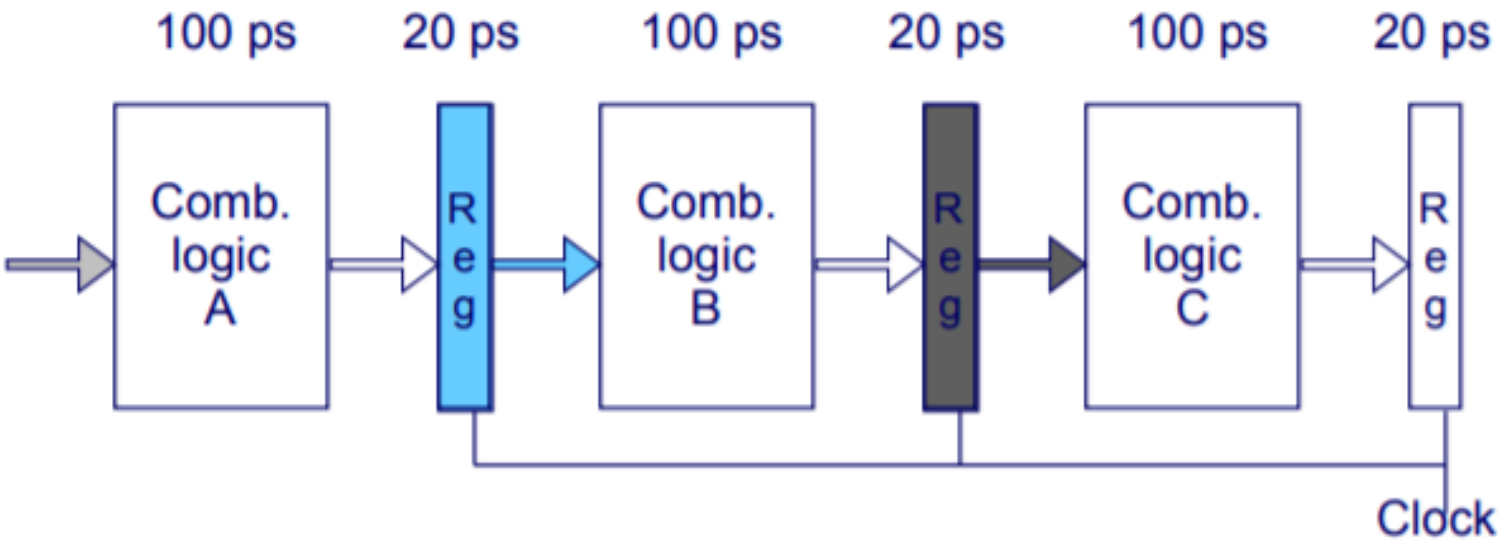
Operación


239
Clock
OP1
OP2
OP3
A
B
C
A
B
C
A
B
C
0
120
240
360
480
640
Time

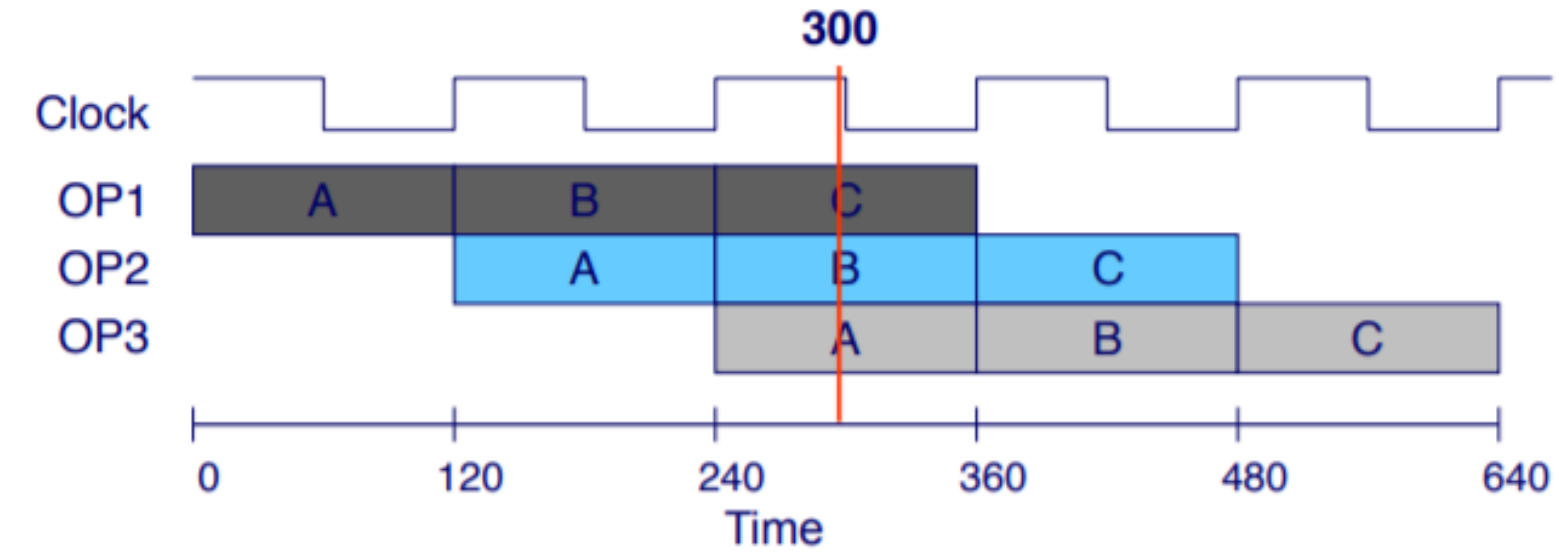

100 ps
20 ps
100 ps
20 ps
100 ps
20 ps
Comb. logic A
Reg
Comb. logic B
Reg
Comb. logic C
Reg
Clock

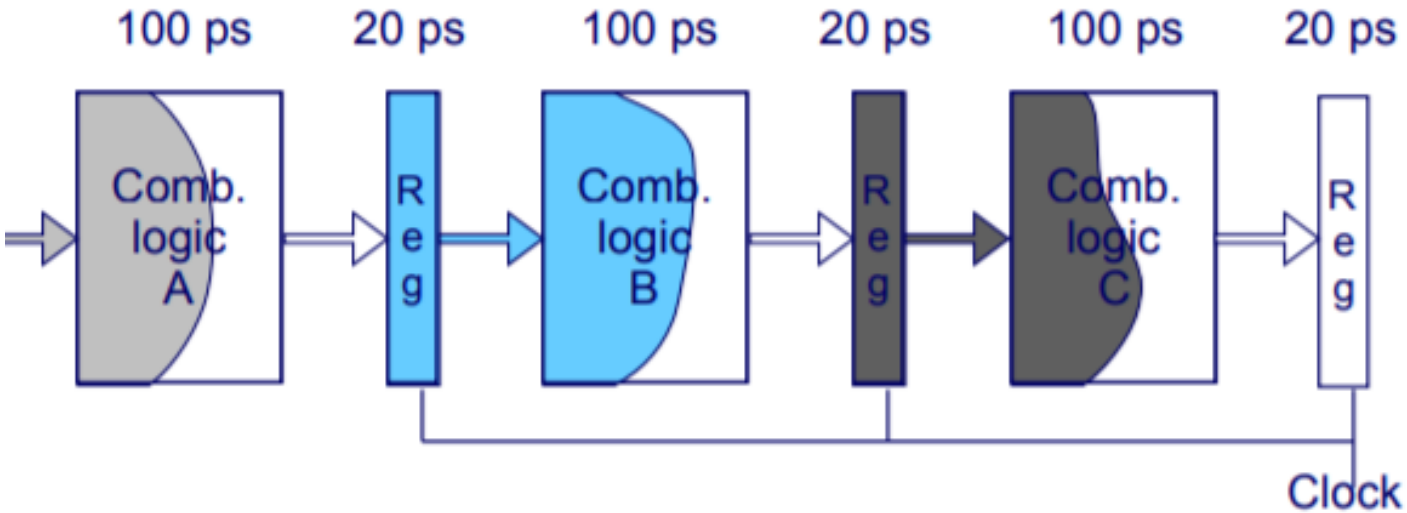
Operación


241
Clock
OP1
OP2
OP3
A
B
C
A
B
C
A
B
C
0
120
240
360
480
640
Time

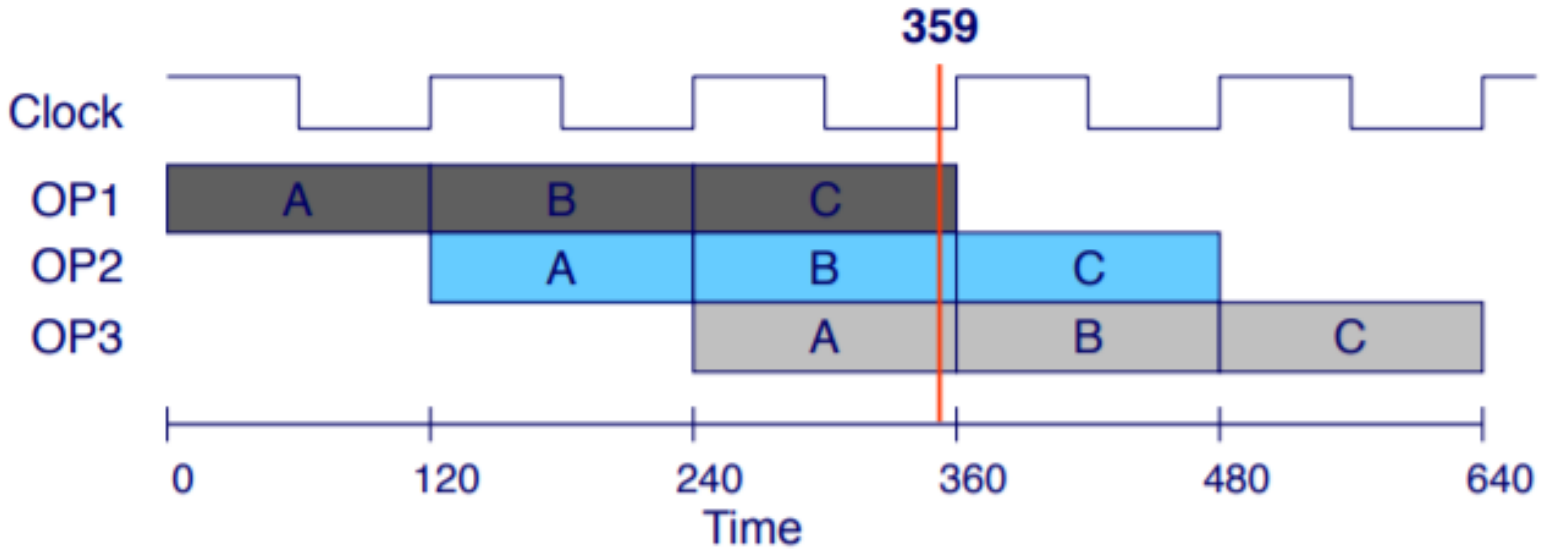

100 ps
20 ps
100 ps
20 ps
100 ps
20 ps
Comb. logic A
Reg
Comb. logic B
Reg
Comb. logic C
Reg
Clock

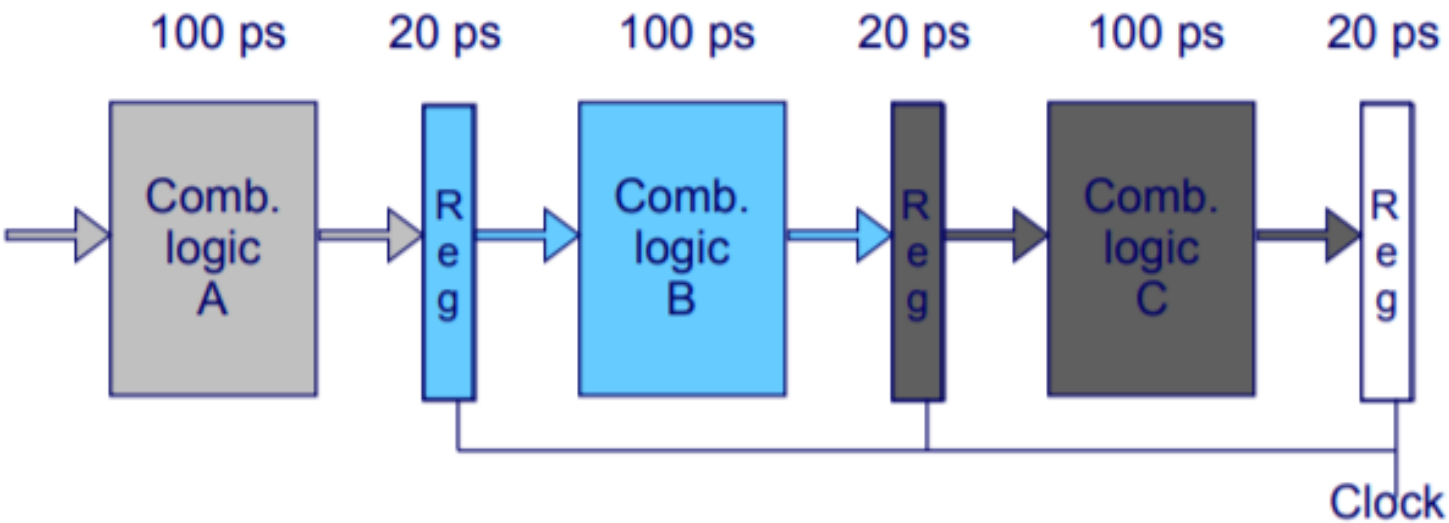
Operación


300
Clock
OP1
OP2
OP3
A
B
C
A
B
C
A
B
C
0
120
240
360
480
640
Time

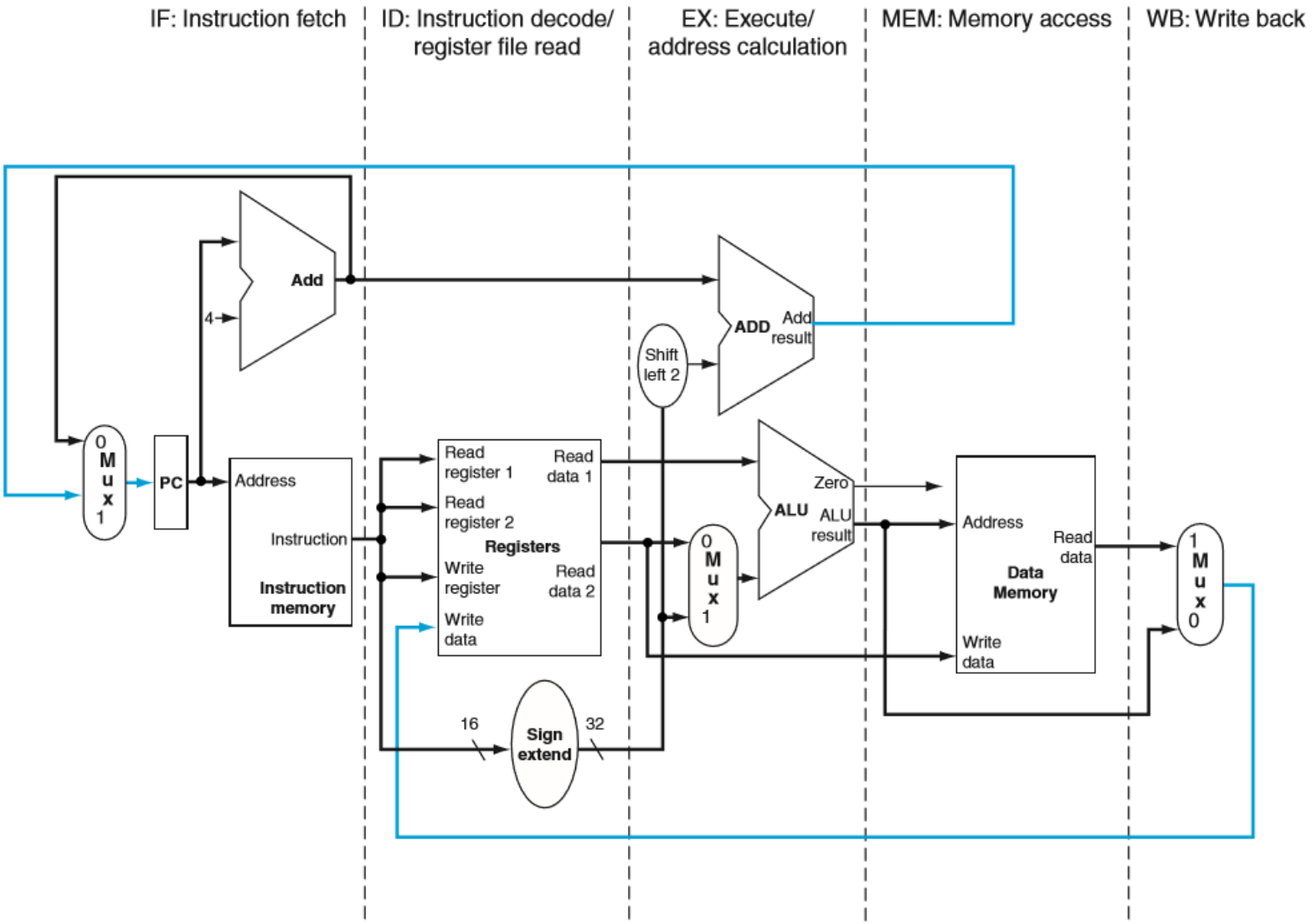

100 ps
20 ps
100 ps
20 ps
100 ps
20 ps
Comb. logic A
Reg
Comb. logic B
Reg
Comb. logic C
Reg
Clock

Operación

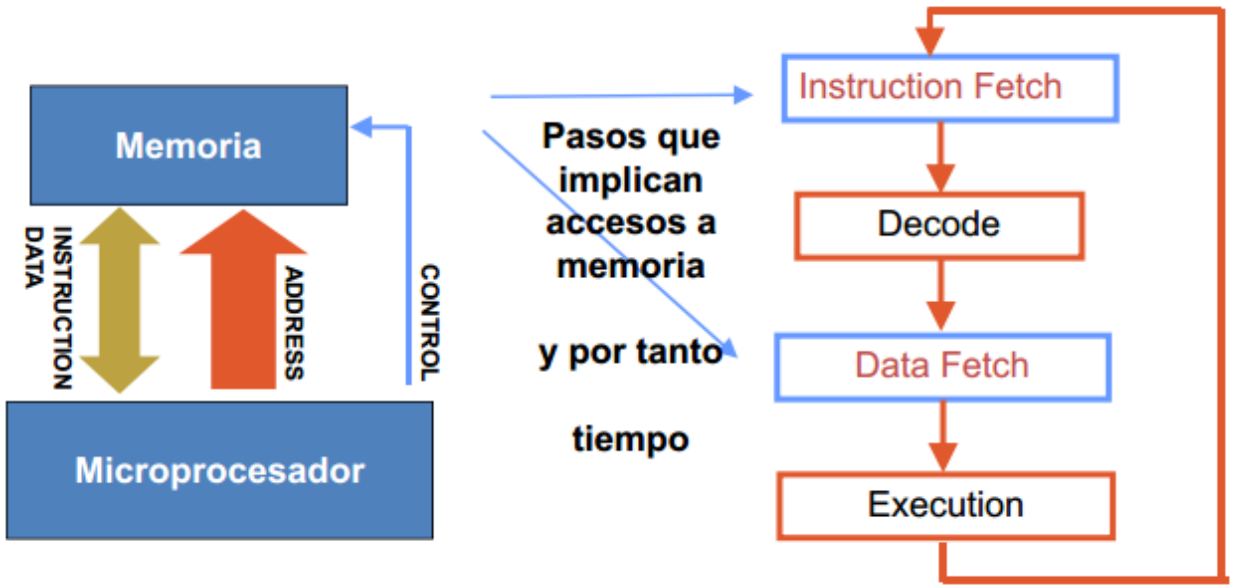

359
Clock
OP1
OP2
OP3
A
B
C
A
B
C
A
B
C
0
120
240
360
480
640
Time


100 ps
20 ps
100 ps
20 ps
100 ps
20 ps
Comb. logic A
Reg
Comb. logic B
Reg
Comb. logic C
Reg
Clock

Concepto

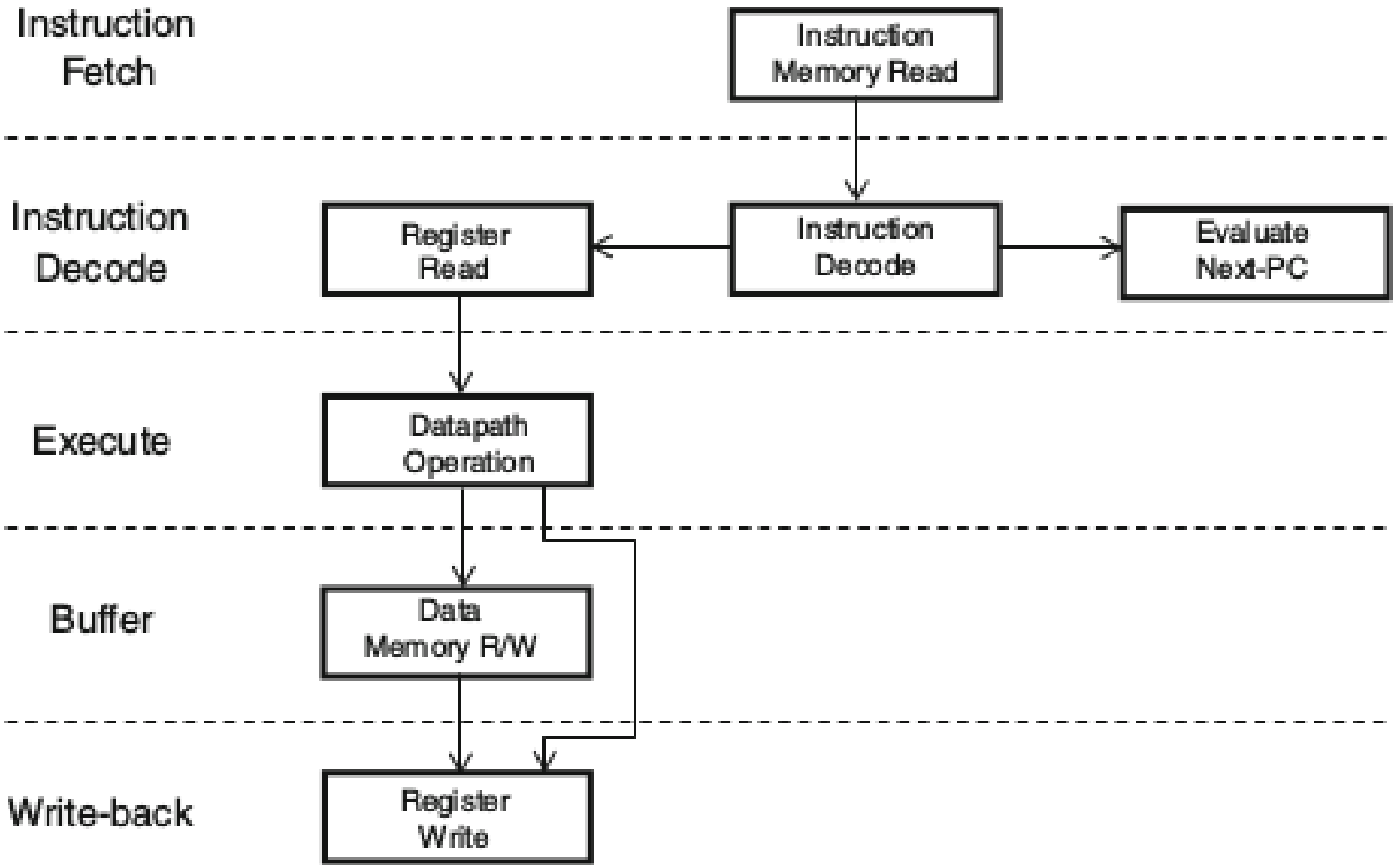

IF: Instruction fetch
ID: Instruction decode/ register file read
EX: Execute/ address calculation
MEM: Memory access
WB: Write back
Add
4
M u x
0
1
PC
Address
Instruction
Instruction memory
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Shift left 2
ADD
Add result
ALU
Zero
ALU result
Address
Read data
Data Memory
Write data
16
Sign extend
32

Ciclo de instrucción

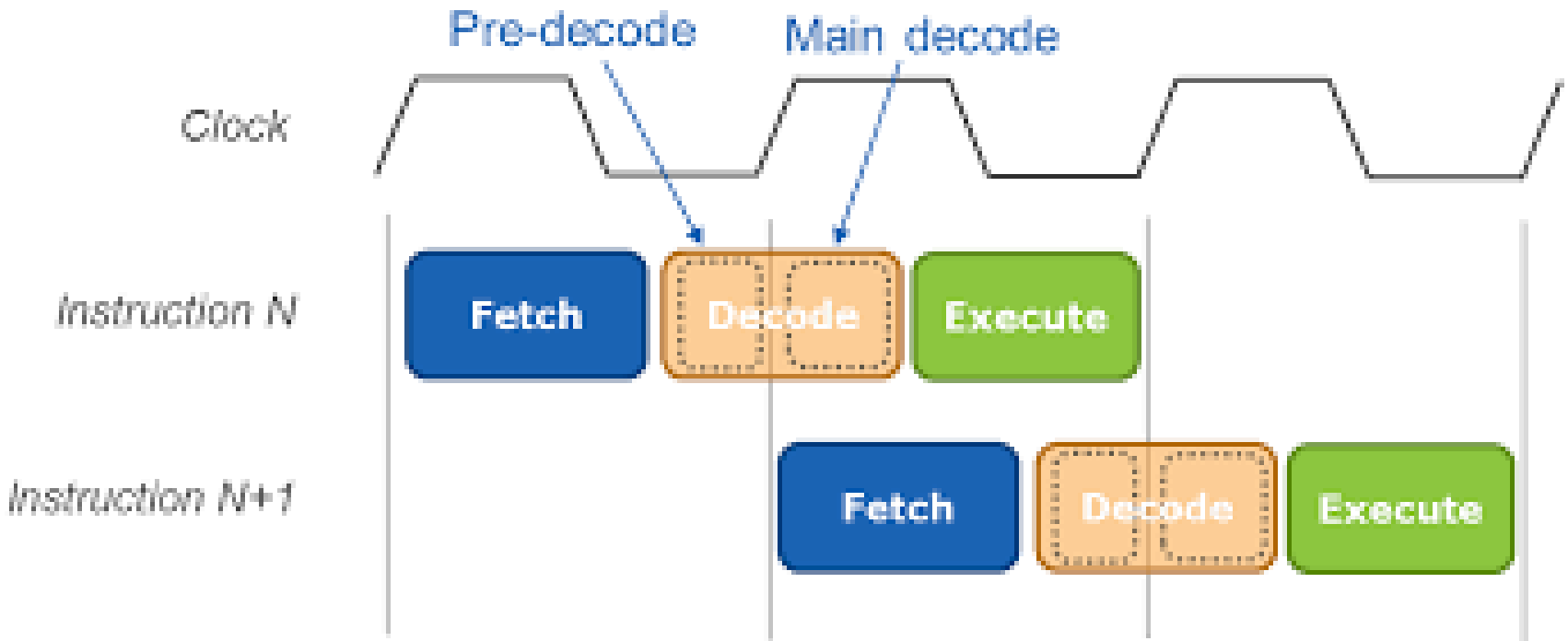


- Buscar la instrucción en la memoria principal
- Decodificar la instrucción
- Ejecutar la instrucción
- Almacenar o guardar resultados

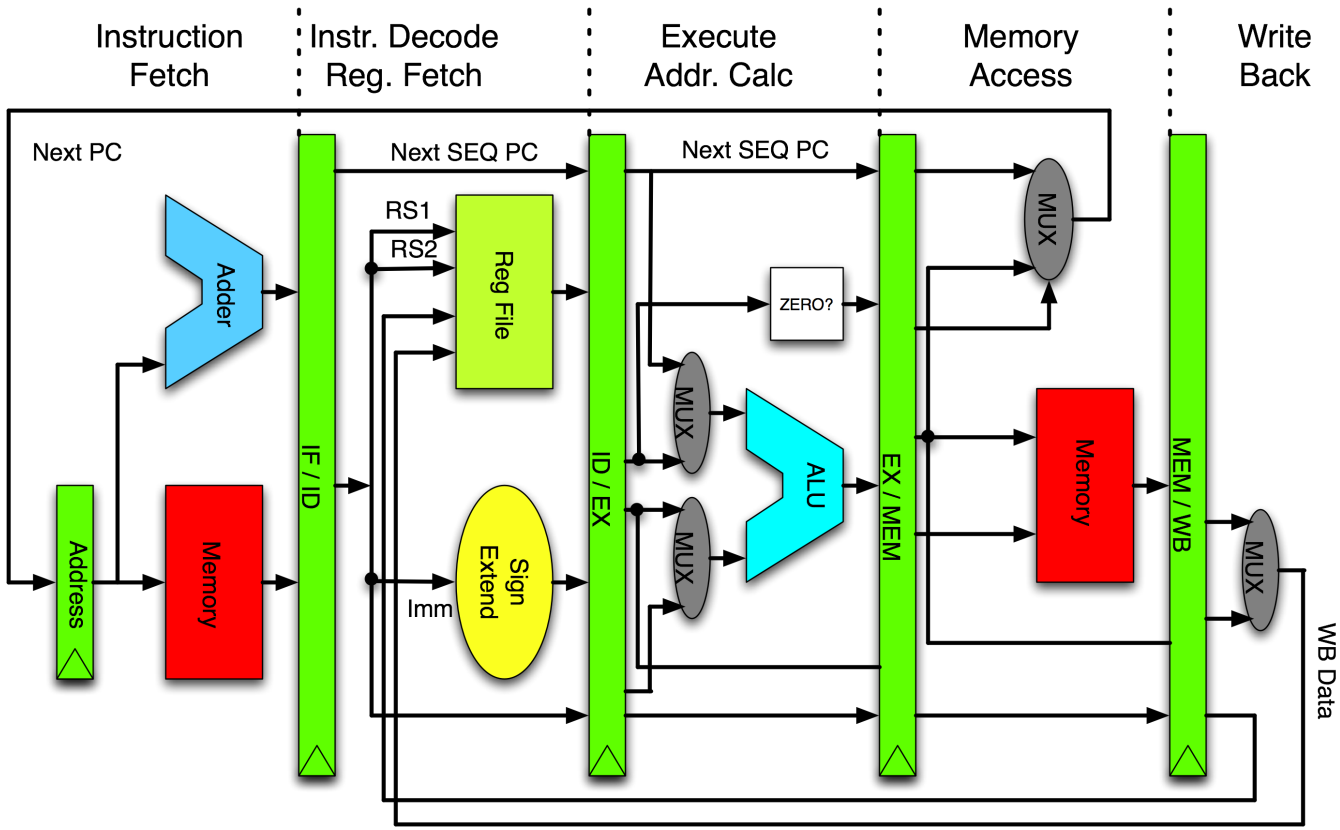
RISC Pipeline



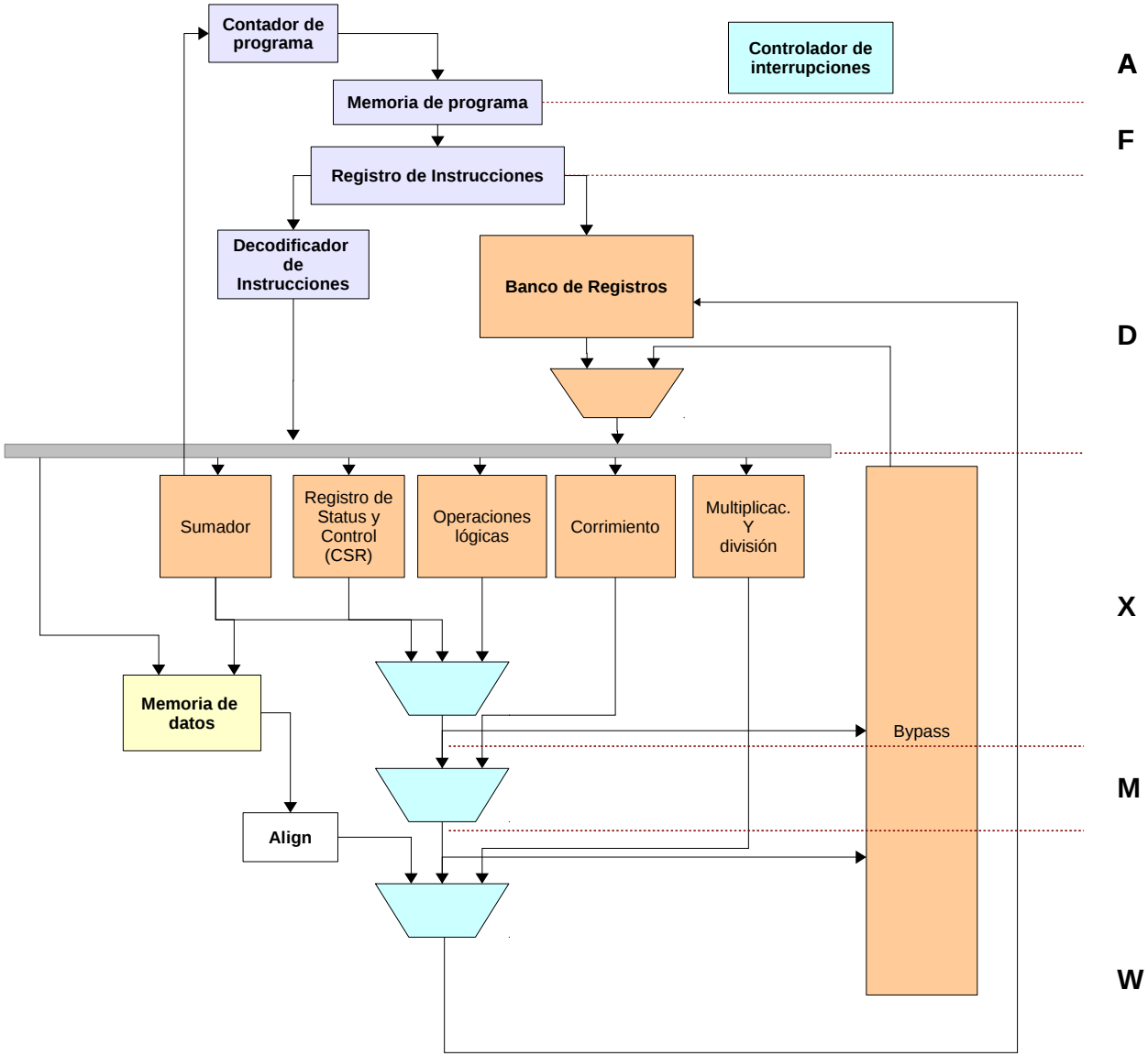
Pipeline Cortex M0 3 Ciclos



MIPS Pipeline 5 ciclos



LM32 Pipeline



Pipeline

A *Address*: Se calcula la dirección de la instrucción a ser ejecutada y es enviada al registro de instrucciones.

F *Fetch*: La instrucción se lee de la memoria.

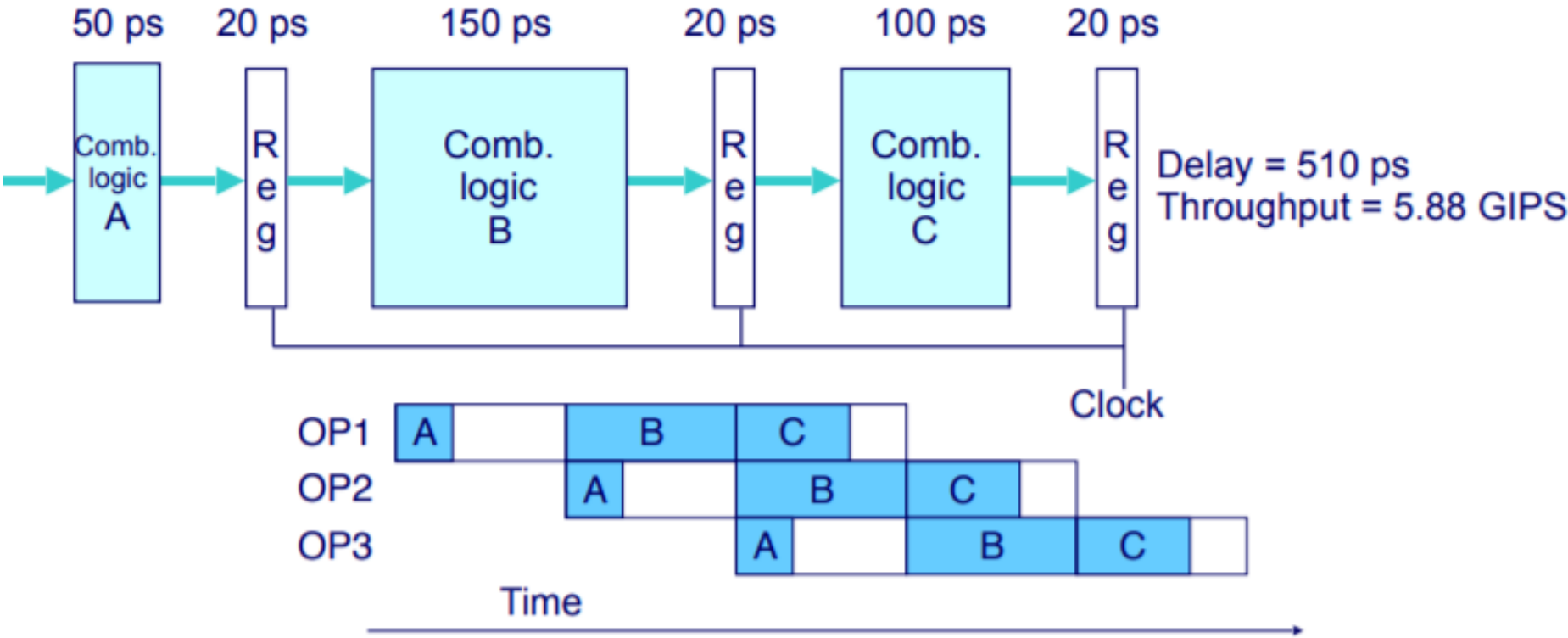
D *Decode*: Se decodifica la instrucción y se toman los operandos del banco de registros o tomados del bypass.

X *Execute*: Se realiza la operación especificada por la instrucción. Para instrucciones simples (sumas y operaciones lógicas), la ejecución finaliza en esta etapa, y el resultado se hace disponible para el bypass.

M *Memory*: Para instrucciones más complejas como acceso a memoria externa, multiplicación, corrimiento, división, es necesaria otra etapa.

D *Write back*: Los resultados producidos por la instrucción son escritas al banco de registros.

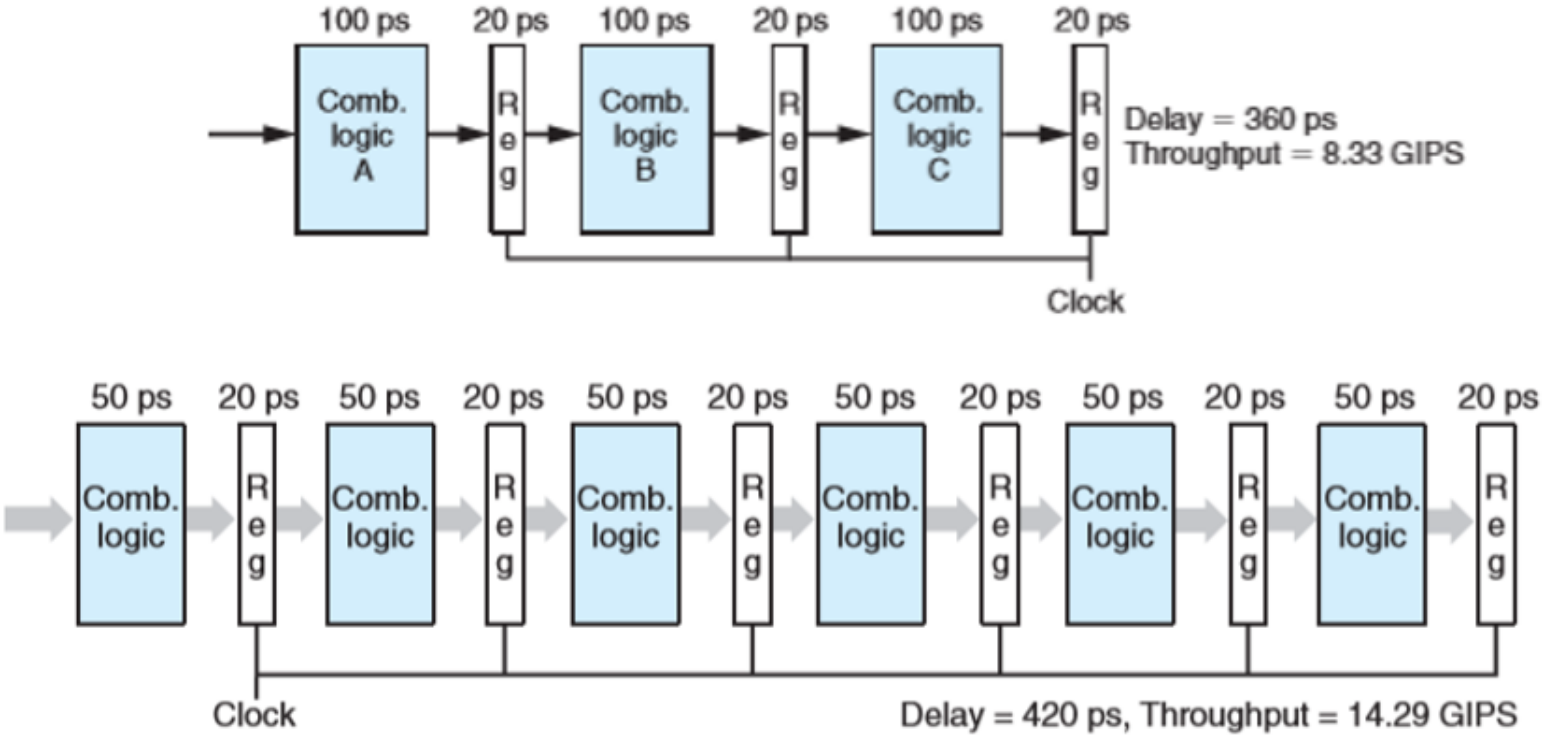
Limitaciones de Pipeline, no uniforme



Rendimiento limitado por la etapa más lenta.

Otras etapas permanecen inactivas la mayor parte del tiempo

Limitaciones de Pipeline, ¿menor rendimiento a mayor pipeline?



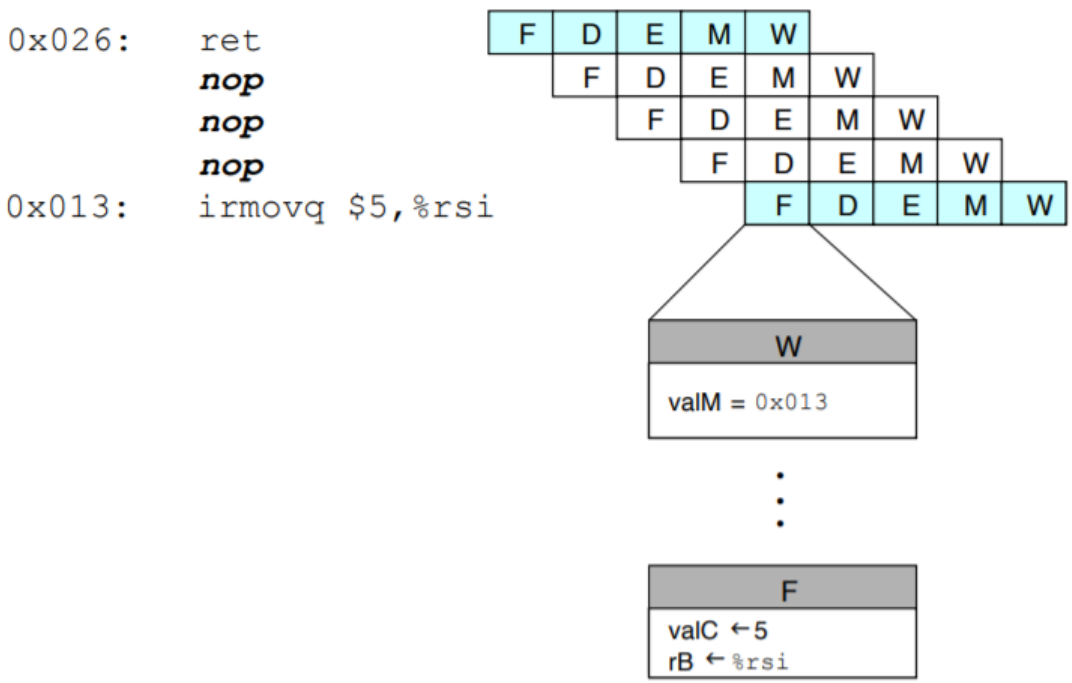
Aumenta el tiempo de almacenamiento

3 Estado = 16.67 %, 6 estados =28.57 %

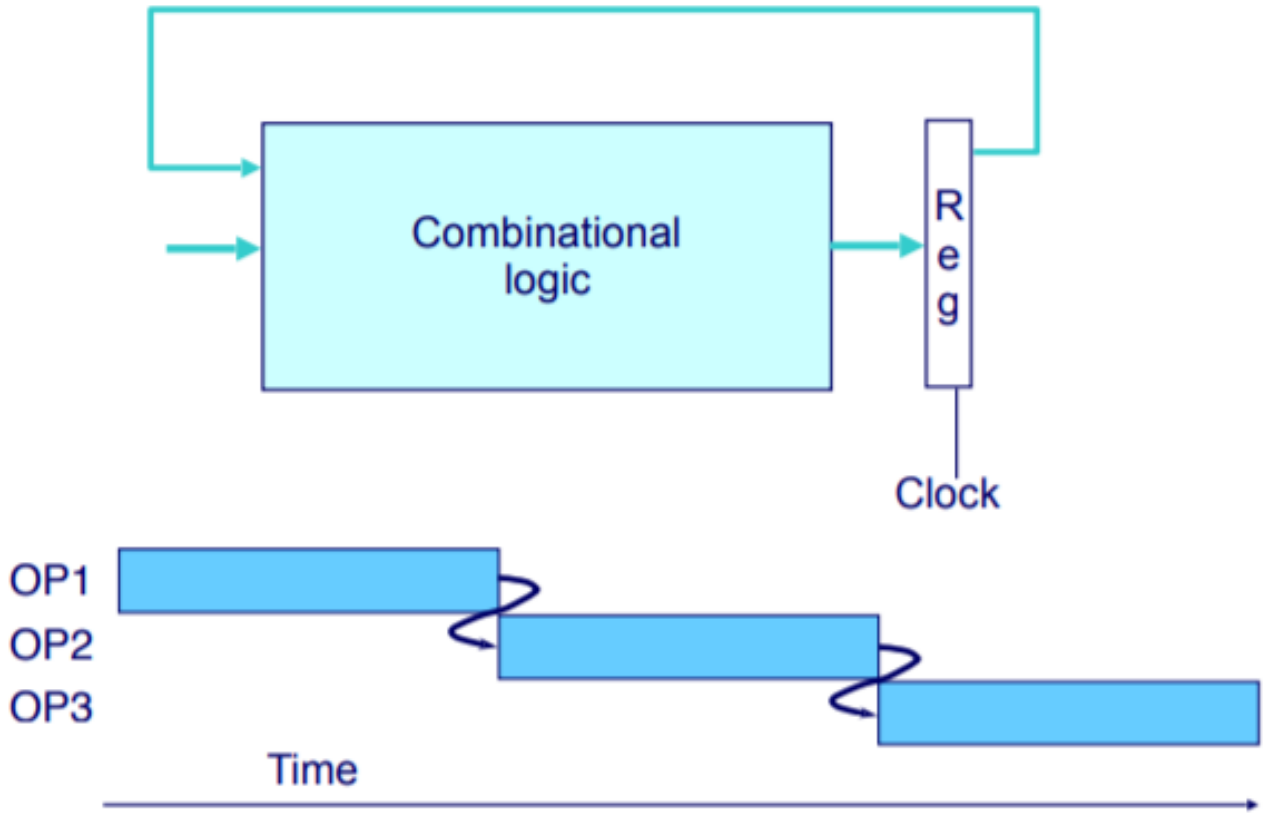
Limitaciones de Pipeline, Control de Dependencias

El resultado de la instrucción A determina si la instrucción B debe ejecutarse o no.

1. Insertar instrucciones que no dependen del efecto de la instrucción anterior.
2. El compilador inserte nops, que también toman espacio de memoria
3. Mecanismos en Hardware.



Dependencia de datos



Ejemplo

- 0x000: movi $10,$R1
- 0x004: movi $3,$R2
- 0x008: add $R1,$R2

En el ciclo 4, los registros 2 y 3 no se han actualizado.

Data Hazards

El riesgo ocurre cuando el resultado no se retro-alimenta a tiempo para la siguiente operación

Program

```
start: mov   r0, #5
       ldr   r1, [r0]
       add   r2, r1, #3
       mov   r3, #0
       nop
```

Cycle	Fetch	Decode	Execute	Buffer	Writeback
0	mov r0, #5				
1	ldr r1,[r0]	mov r0, #5			
2	add r2,r1,#3	ldr r1,[r0]	mov r0, #5		
3	mov r3,#0	add r2,r1,#3	ldr r1,[r0]	mov r0, #5	
4	wait	wait	wait	ldr r1,[r0]	mov r0, #5
5		mov r3,#0	add r2,r1,#3	unused	ldr r1,[r0]
6			mov r3,#0	add r2,r1,#3	unused
7				mov r3,#0	add r2,r1,#3
8					mov r3,#0

Structural Hazards

Program

```
mov   r0, #5
ldmia r0, {r1,r2}
add   r4, r1, r2
add   r4, r4, r3
```

Cycle	Fetch	Decode	Execute	Buffer	Writeback
0	mov r0, #5				
1	ldmia r0,{r1,r2}	mov r0, #5			
2	add r4,r1,r2	ldmia r0,{r1,r2}	mov r0, #5		
3	wait	wait	ldmia r0,{r1,r2}	mov r0, #5	
4	add r4,r4,r3	add r4,r1,r2	ldmia r0,{r1,r2}	*load r1*	mov r0, #5
5		add r4,r4,r3	add r4,r1,r2	*load r2*	*update r1*
6			add r4,r4,r3	add r4,r1,r2	*update r2*
7				add r4,r4,r3	add r4,r1,r2
8					add r4,r4,r3

Referencias bibliográficas

1. Computer Organization and Design 5th Edition Patterson Hennessy
2. Computer Systems: A Programmer'sPerspective, 3th Edition, Bryant and O'Hallaron's