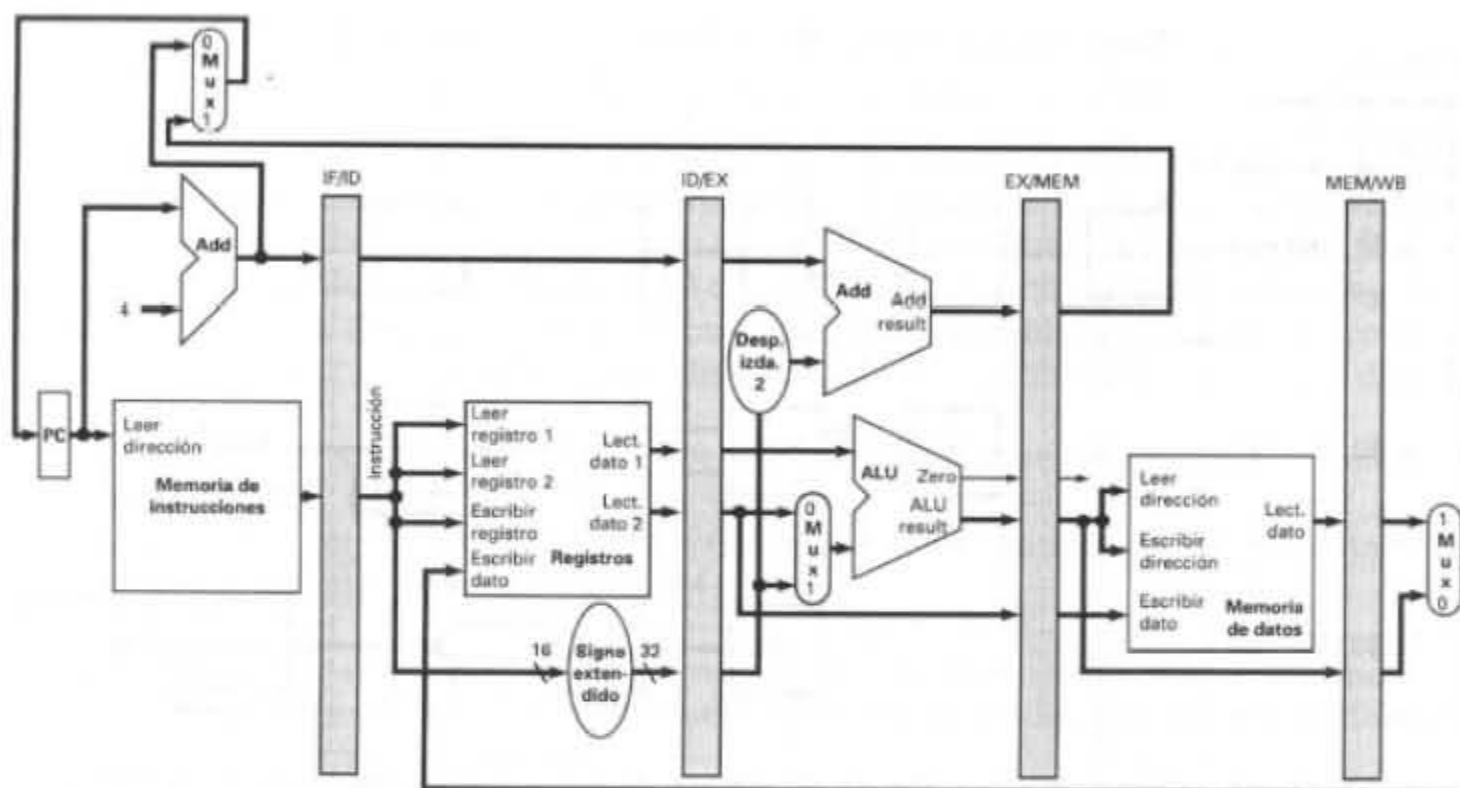


# ARQUITECTURAS AVANZADAS DE PROCESADORES

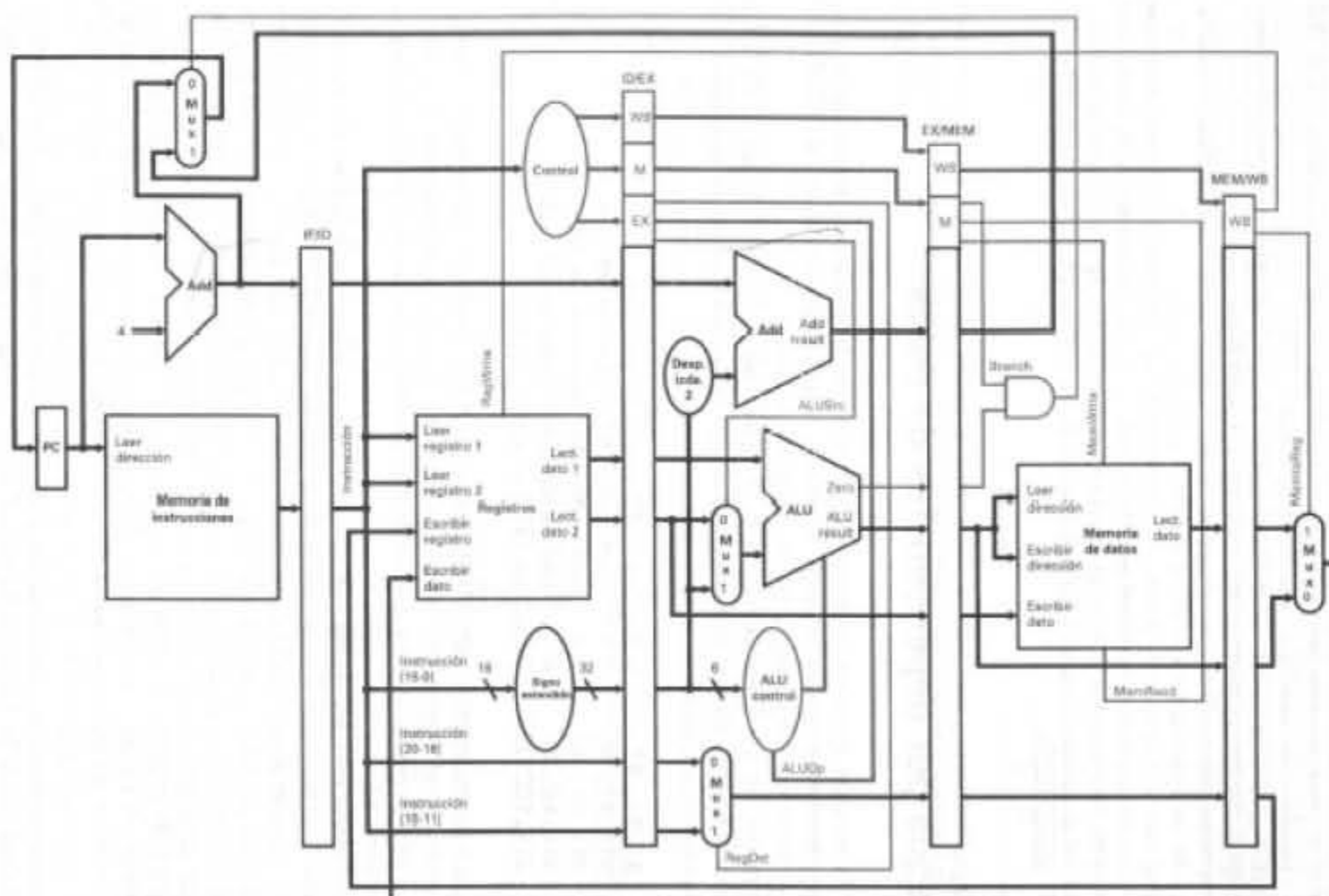
## Mejora del rendimiento con la segmentación

### HOJA DE PROBLEMAS N° 1

- 1.- Si el tiempo para una operación de ALU puede reducirse un 25%, ¿afectará a la mejora obtenida por la segmentación? ¿Y si el tiempo de dicha operación aumenta un 25%?
- 2.- Supongamos un programa con  $10^6$  instrucciones, en el que cada instrucción toma 100 ps. ¿Cuánto tiempo será necesario para ejecutarlo en un procesador no segmentado? Con un procesador con 20 etapas de segmentación, perfectamente segmentado, ¿cuál será la mejora respecto al anterior?
- 3.- Para cada registro de segmentación de la figura, rotular cada parte del registro de segmentación por el nombre del valor que se carga en el registro. Determinar la longitud de cada campo en bits y la longitud total del registro de segmentación.



- 4.- Mostrar las anchuras adicionales de los registros de segmentación de la figura.



# ARQUITECTURAS AVANZADAS DE PROCESADORES

## Mejora del rendimiento con la segmentación

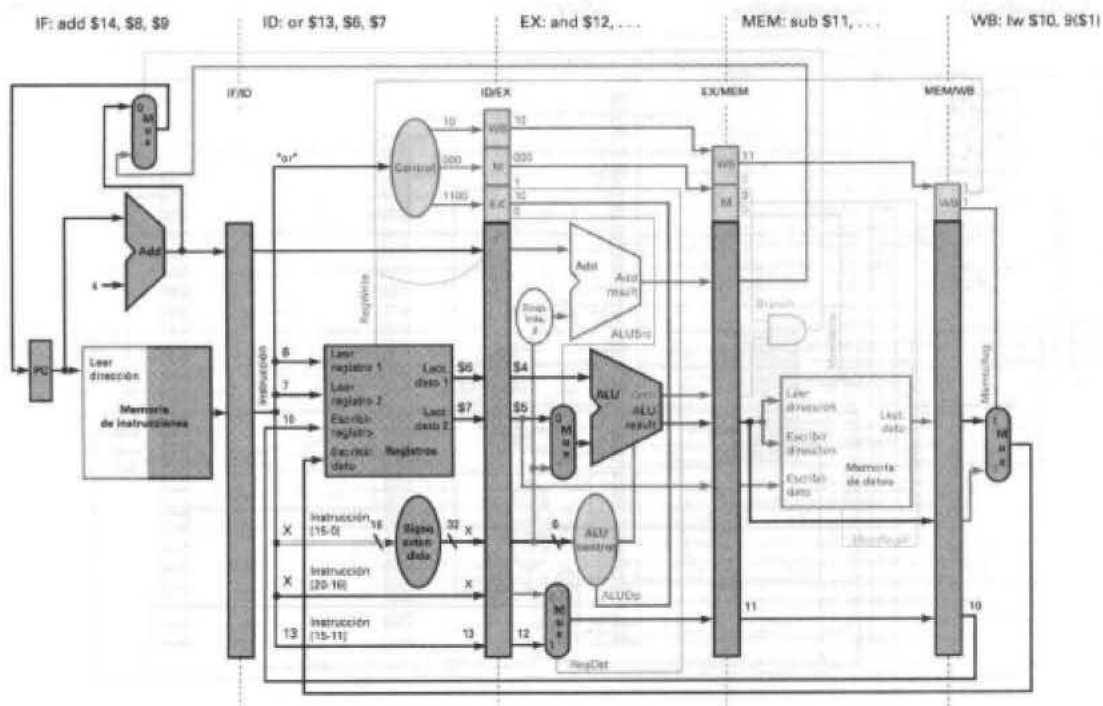
### HOJA DE PROBLEMAS Nº 2

5.- Utilizando la figura siguiente y las instrucciones que se están ejecutando, determinar el valor de cada campo en los cuatro registros de segmentación en el ciclo que se muestra.

**lw \$10,9(\$1)**  
**sub \$11,\$2,\$3**  
**and \$12,\$4,\$5**  
**or \$13,\$6,\$7**  
**add \$14,\$8,\$9**

Suponer que el estado inicial de la máquina era:

- PC tenía el valor 500<sub>diez</sub>, dirección de la instrucción lw.
- Cada registro tenía el valor inicial 10<sub>diez</sub> más el número de registro (el registro \$8 tenía el valor 18<sub>diez</sub>).
- Cada palabra de memoria accedida como dato tenía el valor inicial 1000<sub>diez</sub> más la dirección del byte de la palabra (Memoria[8] tenía el valor inicial 1008<sub>diez</sub>).



6.- Suponiendo un camino de datos segmentado con unidad de detección de riesgos y unidad de anticipación, determinar qué registros habrán sido leídos y cuáles escritos en el quinto ciclo de reloj de la ejecución de:

- |    |                        |    |                        |
|----|------------------------|----|------------------------|
| a) | <b>add \$2,\$3,\$1</b> | b) | <b>add \$2,\$3,\$1</b> |
|    | <b>sub \$4,\$3,\$5</b> |    | <b>sub \$4,\$3,\$5</b> |
|    | <b>add \$5,\$3,\$7</b> |    | <b>add \$8,\$2,\$6</b> |
|    | <b>add \$7,\$6,\$1</b> |    | <b>add \$7,\$6,\$1</b> |
|    | <b>add \$8,\$2,\$6</b> |    | <b>add \$5,\$3,\$7</b> |

7.- Considerando el programa del ejercicio anterior, explicar qué hace la unidad de anticipación durante el quinto ciclo de la ejecución.

8.- Siguiendo con el mismo programa de los ejercicios anteriores, explicar qué hace la unidad de detección de riesgos durante el quinto ciclo.



# ARQUITECTURAS AVANZADAS DE PROCESADORES

## Mejora del rendimiento con la segmentación

HOJA DE PROBLEMAS N° 3

9.- Tenemos un programa con 1000 instrucciones con la forma “lw, add, lw, add...” La instrucción add depende únicamente de la instrucción lw previa. La instrucción lw depende únicamente de la instrucción add previa. Si el programa se ejecuta en un camino de datos segmentado con unidad de detección de riesgos y unidad de anticipación, ¿cuál sería el actual CPI? ¿Y sin unidad de anticipación?

10.- Consideremos un camino de datos segmentado sin anticipación ni detenciones. Se pueden añadir instrucciones **nop**. Reescribir el siguiente código insertando el mínimo número de instrucciones nop, teniendo la posibilidad de reordenar las instrucciones:

```
mov $5,$0
Sum: lw $10,1000($5)
addu $5,$5,$10
addiu $20,$20,-4
bne $20,$0,Sum
```

6.- Suponiendo un camino de datos segmentado con unidad de detección de riesgos y unidad de anticipación, determinar qué registros habrán sido leídos y cuáles escritos en el quinto ciclo de reloj de la ejecución de:

a)     add \$2,\$3,\$1  
          sub \$4,\$3,\$5  
          add \$5,\$3,\$7  
          add \$7,\$6,\$1  
          add \$8,\$2,\$6

b)     add \$2,\$3,\$1  
          sub \$4,\$3,\$5  
          add \$8,\$2,\$6  
          add \$7,\$6,\$1  
          add \$5,\$3,\$7

b) SIN R/W SIMULTANEA:

1.	add \$2				
2.	sub \$4	add \$2			
3.	add \$8	sub \$4	add \$2		
4.	add \$7	add \$8	sub \$4	add \$2	
5.	add \$7	add \$8	nop	sub	add \$2
6.	add \$7	add \$8	nop	nop	sub
7.	add \$5	add \$7	add \$8	nop	nop
8.	-	add \$5	add \$7	add \$8	nop
9.	-	add \$5	nop	add \$7	add \$8
10.	-	add \$5	nop	nop	add \$7
11.	-	add \$5	nop	nop	nop
12.	-	-	add \$5	nop	nop

CON R/W SIMULTANEA:

1.	add \$2				
2.	sub \$4	add \$2			
3.	add \$8	sub \$4	add \$2		
4.	add \$7	add \$8	sub \$4	add \$2	
5.	add \$7	add \$8	nop	sub	add \$2
6.	add \$5	add \$7	add \$8	nop	sub
7.	-	add \$5	add \$7	add \$8	nop
8.	-	add \$5	nop	add \$7	add \$8
9.	-	add \$5	nop	nop	add \$7
10.	-	-	add \$5	nop	nop

The diagram illustrates a complex computer architecture with the following components and connections:

- PC Queue**: Receives the **PC** value and feeds into the **Instruction memory**.
- Instruction memory**: Outputs to the **IO/EX Mem/Read** unit.
- IO/EX Mem/Read**: A central unit that interfaces with the **Hazard Detection unit**, **IO/EX**, **EXMEM**, and **MEMWB**.
- Hazard Detection unit**: Receives signals from the **IO/EX Mem/Read** and the **PC Queue**.
- IO/EX**: A unit that receives data from the **IO/EX Mem/Read** and the **Registers**, and outputs to the **EXMEM**.
- Registers**: Receives data from the **IO/EX Mem/Read** and the **IO/EX**, and outputs to the **IO/EX**.
- IO/EX**: A unit that receives data from the **IO/EX Mem/Read** and the **Registers**, and outputs to the **EXMEM**.
- EXMEM**: Receives data from the **IO/EX Mem/Read** and the **IO/EX**, and outputs to the **MEMWB**.
- MEMWB**: Receives data from the **IO/EX Mem/Read** and the **EXMEM**, and outputs to the **Data memory**.
- Data memory**: Receives data from the **MEMWB** and outputs to the **Forwarding unit**.
- Forwarding unit**: Receives data from the **Data memory** and the **IO/EX Mem/Read**, and outputs to the **PC Queue**.
- ALU**: Receives data from the **IO/EX Mem/Read** and the **IO/EX**, and outputs to the **Forwarding unit**.
- MUX** (Multiplexers): Several multiplexers are used to select data from different sources (Registers, IO/EX, EXMEM, MEMWB, Data memory) and feed them into the **Forwarding unit**.

The diagram is labeled with the number **5** in the bottom right corner.

1. add \$2				
2. sub \$4	add \$2			
3. add \$8	sub \$4	add \$2		
4. add \$7	add \$8	sub \$4	add \$2	
5. add \$5	add \$7	add \$8	sub \$4	add \$2

10.- Consideremos un camino de datos segmentado sin anticipación ni detenciones. Se pueden añadir instrucciones **nop**. Reescribir el siguiente código insertando el mínimo número de instrucciones nop, teniendo la posibilidad de reordenar las instrucciones:

```
mov $5,$0  
Sum: lw $10,1000($5)  
addu $5,$5,$10  
addiu $20,$20,-4  
bne $20,$0,Sum
```

Con W/R simultánea y anticipación

```
mov $5,$0
```

```
Sum: lw $10, 1000($5)
```

```
nop
```

```
addu $5, $5, $10
```

```
addiu $20, $20, -4
```

```
bne $20, $0, Sum
```

Con W/R simultánea

```
mov $5,$0
```

```
nop
```

```
nop
```

```
Sum: lw $10, 1000($5)
```

```
nop
```

```
nop
```

```
addu $5, $5, $10
```

```
addiu $20, $20, -4
```

```
nop
```

```
nop
```

```
bne $20, $0, Sum
```