

## Parte 4: Atascos por WAR, WAW y estructurales.

La etapa EX usa la ALU de sumas y restas de punto fijo. En un programa que solo utiliza instrucciones que pasan por la etapa EX, las instrucciones no se pueden *sobrepasar*, ya que todas tardan el mismo tiempo y pasan por las mismas etapas. En los procesadores modernos, no obstante, existen distintas ALUs para las operaciones de suma/resta (EX), multiplicación (MUL) y división (DIV). Además, las instrucciones de multiplicación y división tardan más ciclos que las de suma. Por este motivo, las instrucciones de suma pueden **comenzar después** que las de multiplicación o división, y **terminar antes**, generando efectivamente una ejecución fuera de orden (out of order execution).

Este tipo de ejecución abre la posibilidad a tres nuevos tipos de atascos: WAR (Write After Read, o **atascar la escritura para que termine la lectura**), WAW (Write After Write, o **atascar la escritura para que termine la otra escritura**) y STR (estructurales, o **atascar una instrucción porque la siguiente etapa o estructura del procesador está siendo utilizada**).

### 1)Atascos WAR y WAW ↗

Los atascos WAR y WAW son la contracara de los RAW. Si bien es mucho más difícil que se produzcan en un programa común, son posibilidades que debe tener en cuenta el procesador. Estos atascos suceden cuando una **instrucción más rápida** se adelante a una **instrucción más lenta**. Los siguientes programas presentan ejemplos minimalistas de tipo de atascos. Responder:

1. Estudiar el código sin simularlo y responder ¿cuál es el programa que tiene WAR y cual WAW?  
Simular los programas para comprobar.
2. En el caso del WAR, ¿cuál es la instrucción lenta y cuál la rápida? ¿Qué registro se quiere leer y escribir?
3. Idem para el caso WAW.

<pre>.code ddiv \$t1, \$t2, \$t3 dadd \$t1, \$t2, \$t3 halt</pre>	<pre>.code dmul \$t1, \$0, \$0 dmul \$t3, \$t1, \$t2 dadd \$t2, \$0, \$0 halt</pre>
---	---

1. El programa A tiene atascos WAW. El programa B tiene atascos WAR  
Las dos escriben \$t1 pero dadd tiene que esperar a que ddiv termine.
2. En el programa A, la instrucción lenta es ddiv (tarda 24 ciclos) y la rápida dadd.  
dadd quiere escribir \$t2 pero dmull que es más lenta, antes tiene que leerla.

### 2)Atascos estructurales (STR) ↗

La ejecución fuera de orden también permite otro tipo de atasco. Los atascos estructurales suceden cuando dos instrucciones quieren acceder al mismo tiempo a la misma etapa del cauce. En el simulador, esto sucede en la etapa MEM, ya que sin importar la ALU que usen las instrucciones, luego del cálculo siempre deben pasar por la etapa MEM. Como las instrucciones rápidas pueden sobrepasar a las lentas, puede suceder que dos o más instrucciones terminen su etapa de ejecución al mismo tiempo, y por ende también quieran pasar a MEM al mismo tiempo.

<pre>.code dmul \$t1, \$0, \$0 nop nop nop</pre>	<pre>nop nops dadd \$t2, \$0, \$0 halt</pre>
--	--

- a) Ejecutar el código anterior y verificar que ocurre un atasco STR. ¿Entre qué instrucciones sucede el atasco? ¿cuál es la instrucción que se atasca? ¿Por qué esa y no la otra?
  - b) Probar agregando un NOP ¿sigue el atasco? ¿y si se quita un NOP? ¿por qué?
- a.- Se atasca en la instrucción dadd ya que coincide la salida de EX a MEM entre dadd (1 ciclo) y dmul (7 ciclos). Ocurre en dadd debido a que es la instrucción mas nueva.
- b.- Sigue habiendo un atasco, pero ahora el incidente ocurre entre nop y dmul, por la misma causa.  
Si se quita un nop, se produce el atasco pero ahora con el halt.

### 3) Análisis de atascos

Los siguientes programas presentan ejemplos naturales de atascos estructurales y WAR. Analizar e identificar dónde pueden ocurrir estos atascos. Ejecutar en el simulador y comprobar el resultado.

```

; Resto: Calcula en $t4 el resto
; de $t1 div $t2
.code
daddi $t1,$0,30 ; a = 30 daddi
$t2,$0,4 ; b = 4
ddiv $t3,$t1,$t2 ; c = a div b = 7 dmul
$t3, $t3, $t3 ; cxb = 7*4 = 28 dsub $t4,
$t1,$t3 ; resto = a-cxb = 2 halt

; factorial: Calcula en $t2
; el factorial de $t1
.code
daddi $t1,$0,5 ; n=5 daddi
$t2,$0,1 ; f=1
loop: dmul $t2,$t2,$t1 ; f=fxn daddi
$t1,$t1,-1 ; n=n-1 bnez $t1, loop
halt

```

#### 4) Etapas y atascos 🚫

Completar las etapas en la ejecución de los siguientes programas, asumiendo forwarding activado.

##### a) Suma y producto

Ciclo	1	2	3	4	5	6	7	8	9	11	11	12	13	14	15	16	17	18
ld \$t1, A(\$0)	IF	ID	EX	ME	WB													
ld \$t2, B(\$0)		IF	ID	EX	ME	WB												
dmul \$t3,\$t1,\$t2			IF	ID														
sd \$t3, MULT(\$0)				IF	ID													
dadd \$t3,\$t1,\$t2					IF													
sd \$t3, SUMA(\$0)						IF									EX	ME	WB	
halt															IF	ID	EX ME WB	

##### b) Promedio

Ciclo	1	2	3	4	5	6	7	8	9	11	11	12	13	14	15	16	17
ld \$t1, suma(\$0)	IF	ID	EX	ME	WB												
ld \$t2, cant(\$0)		IF	ID	EX	ME	WB											
ddiv \$t3,\$t1,\$t2			IF	ID													
sd \$t3, prom(\$0)				IF													
halt					F		ID										

Ciclo	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
ld \$t1, suma(\$0)																
ld \$t2, cant(\$0)																
ddiv \$t3,\$t1,\$t2														ME	WB	
sd \$t3, prom(\$0)														ME	WB	
halt														EX	ME	WB