

Tema 4. Problema 1

```
1. lw r1,0x1ac ; r1 ← M(0x1ac)
2. lw r2,0xc1f ; r2 ← M(0xc1f)
3. add r3,r0,r0 ; r3 ← r0+r0
4. mul r4,r2,r1 ; r4 ← r2*r1
5. add r3,r3,r4 ; r3 ← r3+r4
6. add r5,r0,0x1ac ; r5 ← r0+0x1ac
7. add r6,r0,0xc1f ; r6 ← r0+0xc1f
8. sub r5,r5,#4 ; r5 ← r5 - 4
9. sub r6,r6,#4 ; r6 ← r6 - 4
10. sw (r5),r3 ; M(r5) ← r3
11. sw (r6),r4 ; M(r6) ← r4
```

Ventana centralizada con emisión ordenada

Solo hay una unidad de carga de memoria

RAW con la instrucción (2), lw r2....

INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12	13	14
lw r1, 0x1a	IF	ID	EX	EX										
lw r2, 0x1f	IF	ID			EX	EX								
add r3, r0, r0	IF	ID			EX									
mul r4, r2, r1	IF	ID					EX	EX	EX	EX				
add r3, r3, r4		IF	ID								EX			
add r5, r0, 0x1a		IF	ID								EX			
add r6, r0, 0x1f		IF	ID								EX			
sub r5, r5, #4		IF	ID									EX		
sub r6, r6, #4			IF	ID								EX		
sw (r5), r3			IF	ID									EX	
sw (r6), r4			IF	ID										EX

RAW en r4 con la instrucción (4), mul r4,r2....

RAW en r5 y en r6 con las instrucciones de sumar en r5 y en r6 anteriores.

Solo hay una unidad SW

RAW en r5 con la resta anterior, sub r5,

Comprobar que no se emiten más instrucciones de las posibles y que es ordenada

Ventana de instrucciones centralizada con emisión desordenada

RAW en r4 con la instrucción mul

Gracias a la emisión desordenada

Solo hay una unidad de carga

RAW en r2 con la instrucción 2

INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12
lw r1, 0x1a	IF	ID	EX	EX								
lw r2, 0x1f	IF	ID			EX	EX						
add r3, r0, r0	IF	ID	EX									
mul r4, r2, r1	IF	ID					EX	EX	EX	EX		
add r3, r3, r4		IF	ID								EX	
add r5, r0, 0x1a		IF	ID	EX								
add r6, r0, 0x1f		IF	ID	EX								
sub r5, r5, #4		IF	ID		EX							
sub r6, r6, #4			IF	ID	EX							
sw (r5), r3			IF	ID								EX
sw (r6), r4			IF	ID						EX		

Gracias a la emisión desordenada

Dependencia en r5 con la instrucción (6) add r5...

RAW en r4 con la instrucción mul (4) que produce r4

RAW en r3 con la instrucción sum (5) que produce r3

Debe poder emitir tres instrucciones por ciclo al menos

Estación de reserva con tres líneas para cada unidad funcional y envío ordenado

Una sola unidad de Load

Aunque el envío sea ordenado la ejecución puede ser desordenada porque hay varias estaciones de reserva

ESTACIÓN DE RESERVA	INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LD	lw <i>lw</i> r1, 0x1a	IF	ID	EX												
LD	lw <i>lw</i> r2, 0x1f	IF	ID			EX										
ADD(1)	add <i>add</i> r3, r0, r0	IF	ID	EX												
MULT(1)	mul <i>mul</i> r4, r2, r1	IF	ID							EX						
ADD(2)	add <i>add</i> r3, r3, r4		IF	ID									EX			
ADD(3)	add <i>add</i> r5, r0, 0x1a		IF	ID	EX											
ADD(1)	add <i>add</i> r6, r0, 0x1f		IF	ID	EX											
ADD(3)	swb <i>swb</i> r5, r5, #4		IF	ID		EX										
ADD(1)	swb <i>swb</i> r6, r6, #4			IF	ID	EX										
ST	sw <i>sw</i> (r5), r3			IF	ID									EX		
ST	sw <i>sw</i> (r6), r4			IF	ID											EX

Entre paréntesis, la estación de reserva de la unidad funcional en la que se ejecuta la instrucción
El orden en el que se asignan depende de la política que utilice la unidad de ID (también denominada ID/ISS)

Solo hay una unidad de almacenamiento y el envío es ordenado

Problema 2

1.	lw	r3,0x10a	; r3 \leftarrow M(0x10a)
2.	addi	r2,r0,#128	; r2 \leftarrow r0+128
3.	add	r1,r0,0x0a	; r1 \leftarrow r0+0x0a
4.	lw	r4,0(r1)	; r4 \leftarrow M(r1)
5.	lw	r5,-8(r1)	; r5 \leftarrow M(r1-8)
6.	mult	r6,r5,r3	; r6 \leftarrow r5*r3
7	add	r5,r6,r3	; r5 \leftarrow r6+r3
8	add	r6,r4,r3	; r6 \leftarrow r4+r3
9	sw	0(r1),r6	; M(r1) \leftarrow r5
10.	sw	-8(r1),r5	; M(r1-8) \leftarrow r5
11.	sub	r2,r2,#16	; r2 \leftarrow r2-16

Emisión Ordenada

Una sola unidad de carga

Se tiene que mantener el orden al retirar las instrucciones

INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
lw <i>r3, 0x10a</i>	IF	ID	EX		ROB	WB													
addi <i>r2, r0, #128</i>	IF	ID	EX	ROB		WB													
add <i>r1, r0, 0x0a</i>	IF		ID	EX	ROB	WB													
lw <i>r4, 0(r1)</i>	IF		ID		EX		ROB	WB											
lw <i>r5, -8(r1)</i>		IF		ID			EX		ROB	WB									
modi <i>r6, r5, r3</i>		IF		ID	EX										ROB	WB			
add <i>r5, r6, r3</i>		IF			ID	EX									EX	ROB	WB		
add <i>r6, r4, r3</i>		IF			ID	EX									EX	ROB	WB		
sw <i>0(r1), r6</i>			IF			ID	EX								EX	ROB	WB		
sw <i>-8(r1), r5</i>			IF			ID	EX									EX	ROB	WB	
swb <i>r2, r2, #16</i>			IF				ID	EX								EX	ROB	WB	

Tiene que comprobarse que **no se decodifican, emiten, ni escriben en el ROB más de dos instrucciones por ciclo, ni se retiran más de tres instrucciones por ciclo.**

Se podría prescindir de la etapa ROB (la etapa EX finaliza con la escritura en el ROB además de en la ventana de instrucciones) y en las instrucciones SW se podría prescindir de la etapa ROB

Emisión Desordenada

Una sola unidad de carga

INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
lw <i>r3, 0x10a</i>	IF	ID	EX		ROB	WB												
addi <i>r2, r0, #128</i>	IF	ID	EX	ROB		WB												
addi <i>r1, r0, 0x0a</i>	IF		ID	EX	ROB	WB												
lw <i>r4, 0(r1)</i>	IF		ID		EX		ROB	WB										
lw <i>r5, -8(r1)</i>		IF		ID			EX		ROB	WB								
mul <i>r6, r5, r3</i>		IF		ID					EX						ROB	WB		
addi <i>r5, r6, r3</i>		IF			ID										EX	ROB	WB	
addi <i>r6, r4, r3</i>		IF			ID		EX	ROB									WB	
sw <i>0(r1), r6</i>			IF			ID		EX	ROB								WB	
sw <i>-8(r1), r5</i>			IF			ID										EX	ROB	WB
sub <i>r2, r2, #16</i>			IF				ID	EX		ROB								WB

También en este caso hay que tener en cuenta que no se pueden decodificar, emitir, ni escribir en el ROB, más de dos instrucciones por ciclo (obsérvese que la instrucción `sub r2,r2,#16` debe esperar un ciclo para su etapa ROB por esta razón), ni se pueden retirar más de tres instrucciones por ciclo.

Se podría prescindir de ROB, y de WB en las instrucciones de almacenamiento SW

3. a)

INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
lw <i>r3</i> , 0x10a	IF	ID	EX		ROB	WB										
addi <i>r2</i> , <i>r0</i> , #128	IF	ID	EX	ROB		WB										
add <i>r1</i> , <i>r0</i> , 0x0a	IF	ID	EX	ROB		WB										
lw <i>r4</i> , 0(<i>r1</i>)	IF	ID		EX		ROB	WB									
lw <i>r5</i> , -8(<i>r1</i>)		IF	ID		EX		ROB	WB								
medi <i>r6</i> , <i>r5</i> , <i>r3</i>		IF	ID		EX							ROB	WB			
add <i>r5</i> , <i>r6</i> , <i>r3</i>		IF	ID									EX	ROB	WB		
add <i>r6</i> , <i>r4</i> , <i>r3</i>		IF	ID			EX	ROB							WB		
sw 0(<i>r1</i>), <i>r6</i>			IF	ID			EX	ROB						WB		
sw -8(<i>r1</i>), <i>r5</i>			IF	ID									EX	ROB	WB	

Se decodifican el mismo número de instrucciones que se captan.

No existen limitaciones para el número de instrucciones por ciclo que se emiten, escriben el ROB, y se retiran.

Están disponibles todas las unidades funcionales que se necesiten para que no haya colisiones (riesgos estructurales).

Se podría prescindir de WB en las instrucciones de almacenamiento SW, y también de ROB si se supone que la escritura finaliza con la escritura en el ROB al mismo tiempo que en la ventana de instrucciones

Se reduce el tiempo del multiplicador

INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	12
lw <i>r3, 0x10a</i>	IF	ID	EX		ROB	WB						
addi <i>r2, r0, #128</i>	IF	ID	EX	ROB		WB						
add <i>r1, r0, 0x0a</i>	IF	ID	EX	ROB		WB						
lw <i>r4, 0(r1)</i>	IF	ID		EX		ROB	WB					
lw <i>r5, -8(r1)</i>		IF	ID	EX		ROB	WB					
mult <i>r6, r5, r3</i>		IF	ID			EX			ROB	WB		
add <i>r5, r6, r3</i>		IF	ID						EX	ROB	WB	
add <i>r6, r4, r3</i>		IF	ID			EX	ROB				WB	
sw <i>0(r1), r6</i>			IF	ID			EX	ROB			WB	
sw <i>-8(r1), r5</i>			IF	ID						EX	ROB	WB
swb <i>r2, r2, #16</i>			IF	ID	EX	ROB						WB

$$T(n) = 12 = \text{TLI} + (n - 1) \times \text{CPI} = 6 + (11 - 1) \times \text{CPI}$$

CPI=0.6

Ejercicio 8

En un programa, una instrucción de salto condicional (a una dirección de salto anterior) dada tiene el siguiente comportamiento en una ejecución de dicho programa:

SSNNN SSNSN SNSSS SSN

donde S indica que se produce el salto y N que no. Indique la penalización efectiva que se introduce si se utiliza:

- a) Predicción fija (siempre se considera que se no se va a producir el salto)
- b) Predicción estática (si el desplazamiento es negativo se toma y si es positivo no)
- c) Predicción dinámica con dos bits, inicialmente en el estado (11).
- d) Predicción dinámica con tres bits, inicialmente en el estado (111).

Nota: La penalización por saltos incorrectamente predichos es de 5 ciclos y para los saltos correctamente predichos es 0 ciclos.

Ejercicio 8

Solución:

En el caso de usar predicción fija, se produciría un fallo del predictor cada vez que se tome el salto, tal y como muestra la Tabla siguiente. Por tanto, la penalización total sería de:

$$P_{fijo} = F_{fijo} \times P = 11 \times 5 = 55 \text{ ciclos}$$

PREDICCIÓN FIJA	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
EJECUCIÓN	S	S	N	N	N	S	S	N	S	N	S	N	S	S	S	S	S	N
PENALIZACIÓN	P	P				P	P		P		P		P	P	P	P	P	

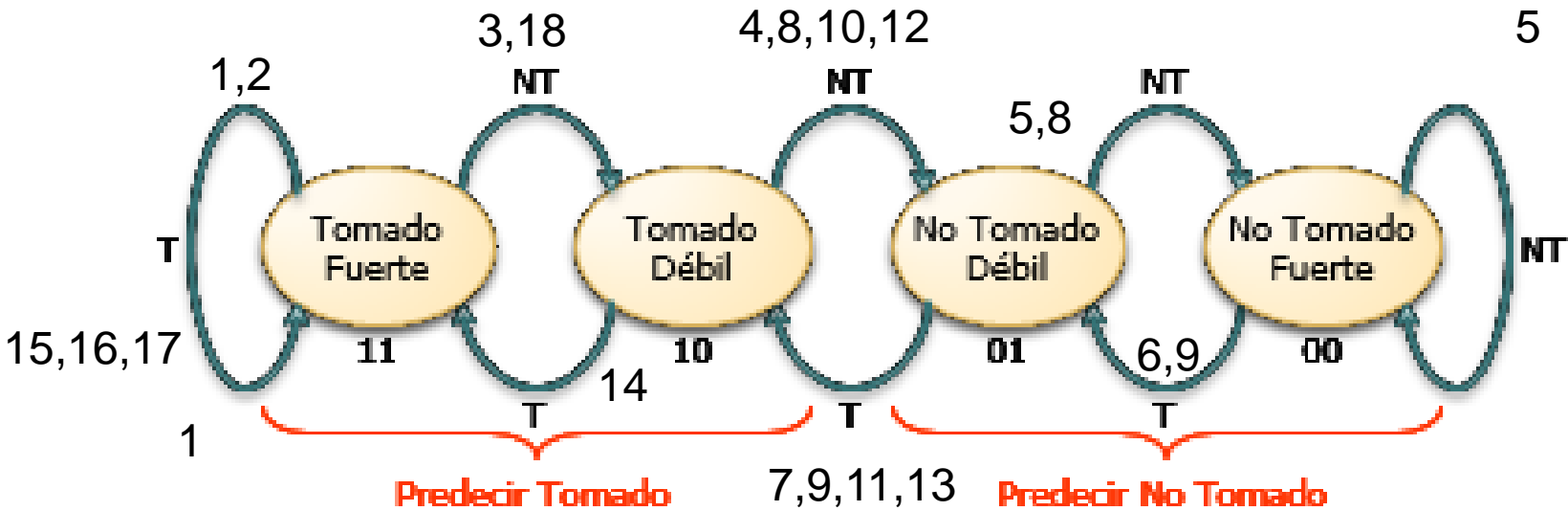
$$P_{estático} = N_{estático} \times P = 7 \times 5 = 35 \text{ ciclos}$$

PREDICCIÓN ESTÁTICA	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
EJECUCIÓN	S	S	N	N	N	S	S	N	S	N	S	N	S	S	S	S	S	N
PENALIZACIÓN			P	P	P			P		P		P						P

Ejercicio 8

$$P_{2 \text{ bits}} = F_{2 \text{ bits}} \times P = 11 \times 5 = 55 \text{ ciclos}$$

ESTADO	11	11	11	10	01	00	01	10	01	10	01	10	01	10	11	11	11	11
PREDICCIÓN DINÁMICA (2 BITS)	S	S	S	S	N	N	N	S	N	S	N	S	N	S	S	S	S	S
EJECUCIÓN	S	S	N	N	N	S	S	N	S	N	S	N	S	S	S	S	S	N
PENALIZACIÓN			P	P		P	P	P	P	P	P	P	P					P



Ejercicio 8

$$P_{3\text{ bits}} = F_{3\text{ bits}} \times P = 10 \times 5 = 50 \text{ ciclos}$$

ESTADO	111	111	111	011	001	000	100	110	011	101	010	101	010	101	110	111	111	111
PREDICCIÓN DINÁMICA (3 BITS)	S	S	S	S	N	N	N	S	S	S	N	S	N	S	S	S	S	S
EJECUCIÓN	S	S	N	N	N	S	S	N	S	N	S	N	S	S	S	S	S	N
PENALIZACIÓN			P	P		P	P	P		P	P	P	P					P

Ejercicio 10

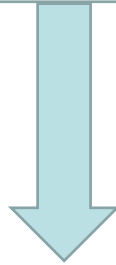
if (A>B) then { X=1;}
else {if (C<D) then X=2; else X=3;}

```
(p1)      lw      r1, a      ; r1 = A
          lw      r2, b      ; r2 = B
          p1, p2  cmp.gt    r1, r2      ; Si A > B p1 = 1 y p2 = 0 (si no, p1 = 0 y p2 = 1)
          addi      r5, r0, #1
          p3      cmp.ne    r0, r0      ; Inicializamos p3 a 0
          p4      cmp.ne    r0, r0      ; Inicializamos p4 a 0
(p2)      lw      r3, c      ; r3 = C
          lw      r4, d      ; r4 = D
          (p2)    p3, p4 cmp.lt    r3, r4      ; Sólo si p2 = 1 p3 o p4 pueden ser 1
          (p3)    addi      r5, r0, #2      ; Se ejecuta si p3 = 1 (y p2 = 1)
          (p4)    addi      r5, r0, #3      ; Se ejecuta si p4 = 1 (y p2 = 1)
          sw      (x), r5      ; Almacenamos el resultado
```

TEMA 4

Ejercicio 9

#	OP1	OP2
1	lw r1, x(r2)	add r10, r11, r12
2		add r13, r10, r14
3	beqz r3, direc	
4	lw r4, 0(r3)	
5	lw r5, 0(r4)	

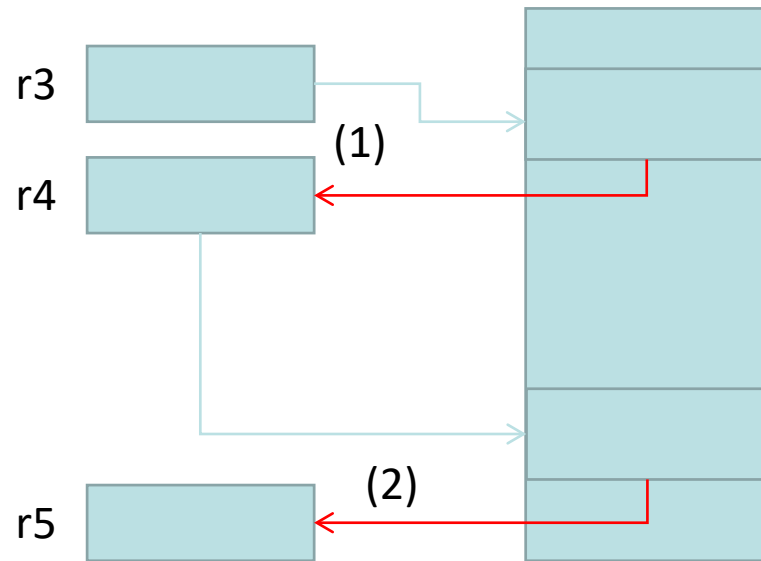


```
lw      r1, x(r2) ; (1)
nop                      ; (2)
beqz    r3, direc ; (3)
lw      r4, 0(r3) ; (4)
lw      r5, 0(r4) ; (5)
```

```

lw      r1, x(r2) ; (1)
nop                               ; (2)
beqz    r3, direc ; (3)
(1) lw   r4, 0(r3) ; (4)
(2) lw   r5, 0(r4) ; (5)

```

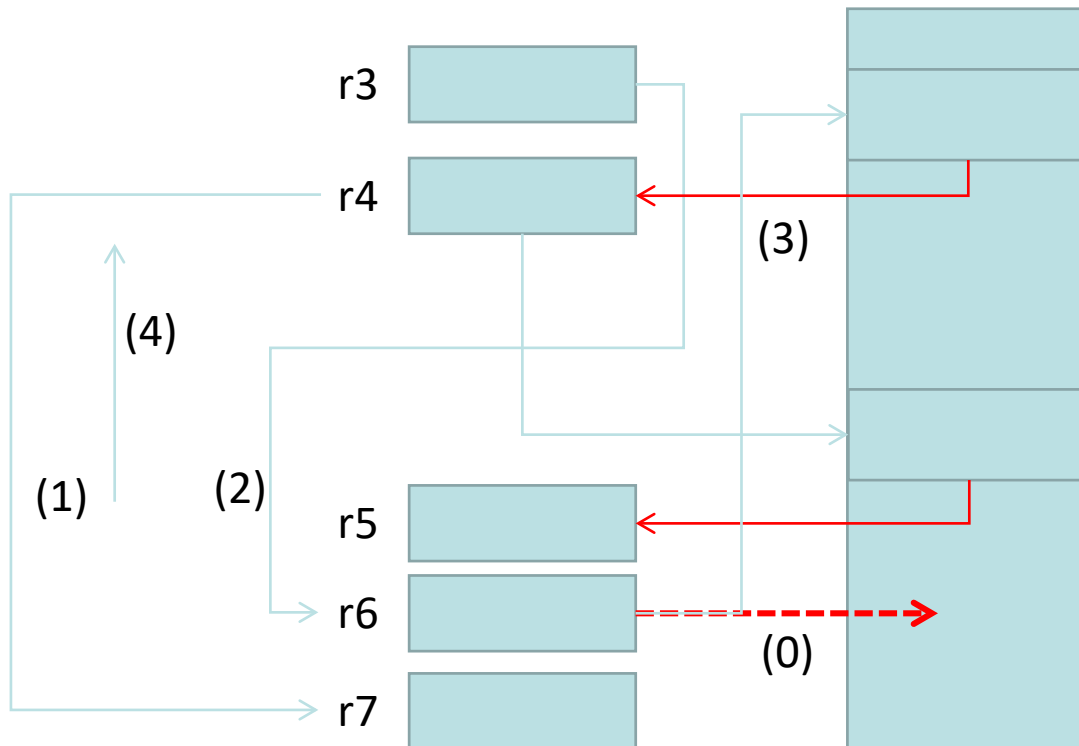


```

addi    r6, r0, #1000      ; Fijamos r6 a una dirección segura
lw      r1, x(r2)
mov     r7, r4              ; Guardamos el contenido original de r4 en r7
cmovnz  r6, r3, r3         ; Movemos r3 a r6 si r3 es distinto de cero
lw      r4, 0(r6)          ; Carga especulativa
cmovz   r4, r7, r3         ; Si r3 es 0, hay que hacer que r4 recupere su valor
beqz    r3, direc
lw      r5, 0(r4) ; Si r3 no es cero, hay que cargar r5

```

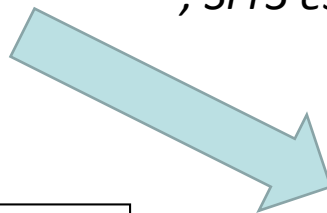

(0)	addi	r6, r0, #1000	; Fijamos r6 a una dirección segura
	lw	r1, x(r2)	
(1)	mov	r7, r4	; Guardamos el contenido original de r4 en r7
(2)	cmovnz	r6, r3, r3	; Movemos r3 a r6 si r3 es distinto de cero
(3)	lw	r4, 0(r6)	; Carga especulativa
(4)	cmovz	r4, r7, r3	; Si r3 es 0, hay que hacer que r4 recupere su valor
	beqz	r3, direc	
	lw	r5, 0(r4)	; Si r3 no es cero, hay que cargar r5



Si r6 no se carga con r3 en **cmovnz** como luego se hace lw hay que asegurarse que esa dirección no genera excepción

cmovz *r5, r8, r3*

; Si r3 es 0 hay que hacer que r5 recupere su valor.


$$lw \quad r5, 0(r4) ; (5)$$

#	OP1	OP2
1	addi r6,r0,#1000	add r10, r11, r12
2	lw r1,x(r2)	add r13, r10, r14
3	cmovnz r6,r3,r3	mov r7,r4
4	lw r4, 0(r3)	mov r8,r5
5	lw r5, 0(r4)	
6	cmovz r4,r7,r3	cmovz r5,r8,r3

Un procesador de 32 bits es capaz de captar y decodificar CUATRO instrucciones por ciclo. Tras ser decodificadas, las instrucciones pasan a una ventana de instrucciones centralizada, desde las que se pueden emitir, de forma DESORDENADA, hasta CUATRO instrucciones por ciclo a las unidades funcionales. Además implementa **adelantamiento (NO ESPECULATIVO)** de Loads a Stores.

El procesador dispone de un buffer de reordenamiento (ROB) que permite implementar la finalización ordenada y el renombramiento y del que se pueden retirar hasta DOS instrucciones por ciclo. Además, contiene DOS unidades de suma (1 ciclo de retardo), UN multiplicador (3 ciclos de retardo), y **UNA unidad de CARGA/ALMACENAMIENTO de memoria** (2 ciclos de retardo).

El procesador tiene, cuatro etapas: IF, ID, EX, WB (no tiene etapa explícita de escritura en el ROB porque se considera que tras el último ciclo de ejecución de una operación, el resultado queda almacenado ya en el ROB

Adelantamiento Load a Stores (No hace falta que sea Especulativo)

		1	2	3	4	5	6	7	8
(1)	add r4,r2,r1	IF	ID	EX	WB				
(2)	<u>sw</u> 0(r3), r4	IF	ID			EX	EX	WB	
(3)	<u>ld</u> r1,16(r3)	IF	ID	EX	EX				WB
(4)	<u>mul</u> r4,r2,r1	IF	ID			EX	EX	EX	WB

Un procesador de 32 bits es capaz de captar y decodificar CUATRO instrucciones por ciclo. Tras ser decodificadas, las instrucciones pasan a una ventana de instrucciones centralizada, desde las que se pueden emitir, de forma DESORDENADA, hasta CUATRO instrucciones por ciclo a las unidades funcionales.

El procesador dispone de un buffer de reordenamiento (ROB) que permite implementar la finalización ordenada y el renombramiento y del que se pueden retirar hasta DOS instrucciones por ciclo. Además, contiene DOS unidades de suma (1 ciclo de retardo), UN multiplicador (3 ciclos de retardo), y **UNA unidad de CARGA/ALMACENAMIENTO de memoria** (2 ciclos de retardo).

El procesador tiene, cuatro etapas: IF, ID, EX, WB (no tiene etapa explícita de escritura en el ROB porque se considera que tras el último ciclo de ejecución de una operación, el resultado queda almacenado ya en el ROB)

Sin adelantamiento Load a Stores

		1	2	3	4	5	6	7	8	9	10	11
(1)	add r4,r2,r1	IF	ID	EX	WB							
(2)	<u>sw</u> 0(r3), r4	IF	ID		EX	EX	WB					
(3)	<u>ld</u> r1,16(r3)	IF	ID				EX	EX	WB			
(4)	<u>mul</u> r4,r2,r1	IF	ID						EX	EX	EX	WB