UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

P3 - Informe Final: El procesador segmentado

Arquitectura de Computadors

Daniel Aagaard Pérez i Pau Martín Nadal

Eva Marin Tordera
I4511

SESIÓN 3

8. Abre el archivo actividad3.s, asegúrate que la configuración de procesador es *Camino de datos -> Segmentado* y el salto es *Salto fijo*, ensambla y dale a ejecutar. Después del salto bne hay 4 instrucciones que no tienen importancia, que están puestas como relleno para que se ejecute todo el bne.

a) ¿Cuántas iteraciones hace el bucle?

El bucle hace 4 iteraciones porque empezamos en @X[12] i en @Y[4] i vamos sumando y restando 4 respectivamente. Entonces el bucle termina cuando los dos valores de $t0 y $t1 tienen el valor @Y[0].

(Valores que nos han ayudado a entender que hace el código.)

$s3 = @X[12];

@Y[4] = $s3;

$t1 = @Y[3];

X[15] + 4 = Y[0]

$t0 = @X[13];

¿Qué hace el código?

i = 12

j = 4

```
while(@x != @y){
        y[j] = x[i];
        i++;
        j--;
}
```

¿En qué posiciones de memoria escribe y qué valores?

[0x10010044] = 0x0000000a

[0x10010048] = 0x00000009

[0x1001004c] = 0x0000000f

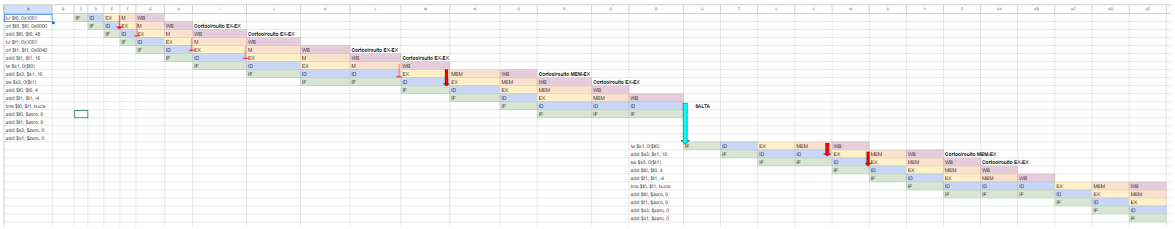
[0x10010050] = 0x0000000a

Segmento de datos

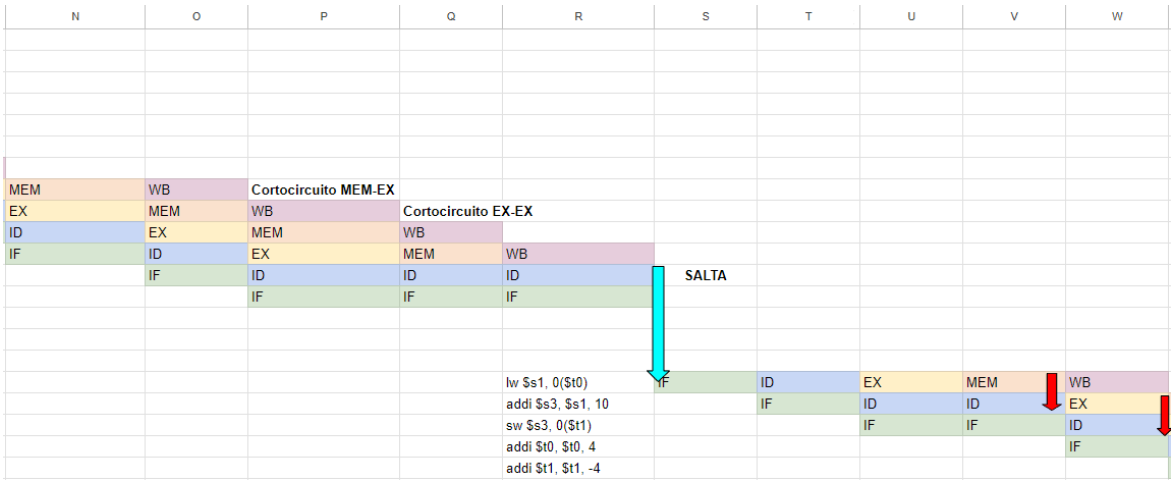
MEMORIA

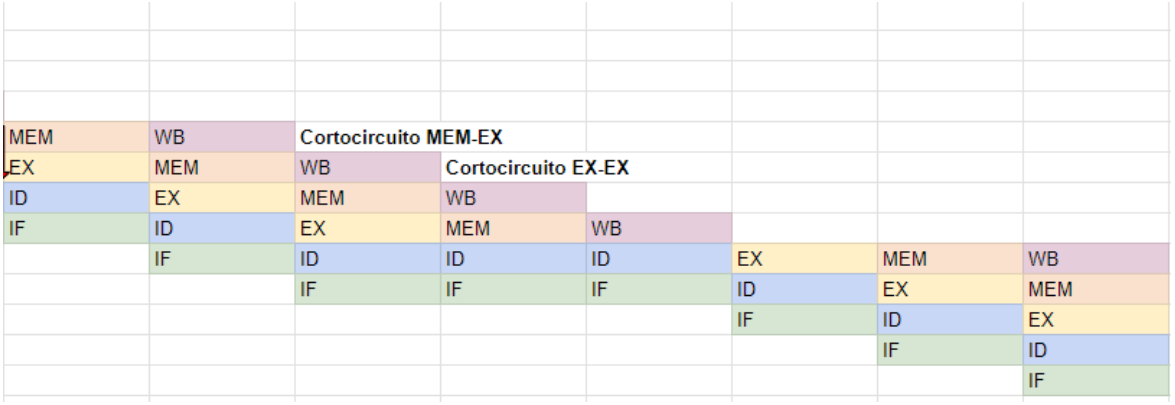
[0x10010000]	0x00000001	0x00000002	0x00000003	0x00000004
[0x10010010]	0xffffffff	0x00000003	0x00000004	0x00000003
[0x10010020]	0xfffffffd	0x00000006	0xffffffff	0x00000004
[0x10010030]	0x00000000	0x00000005	0xffffffff	0x00000000
[0x10010040]	0x00000000	0x0000000a	0x00000009	0x0000000f
[0x10010050]	0x0000000a	0x00000000	0x00000000	0x00000000
[0x10010060]	0x00000000	0x00000000	0x00000000	0x00000000
[0x10010070]	0x00000000	0x00000000	0x00000000	0x00000000

b) Realiza un cronograma de las 2 primeras iteraciones del bucle, mostrando cuando se producen bloqueos (nops hardware o burbujas) y cuando se producen cortocircuitos de datos. A partir de estas dos primeras iteraciones, deduce ¿Cuántos ciclos tarda la ejecución total? ¿Coincide con lo que te da el simulador?



lui $t0, 0x1001	IF	ID	EX	M	WB							
ori $t0, $t0, 0x0000		IF	ID	EX	M	WB	Cortocircuito EX-EX					
addi $t0, $t0, 48			IF	ID	EX	M	WB	Cortocircuito EX-EX				
lui $t1, 0x1001				IF	ID	EX	M	WB				
ori $t1, $t1, 0x0040					IF	ID	EX	M	WB	Cortocircuito EX-EX		
addi $t1, $t1, 16						IF	ID	EX	M	WB	Cortocircuito EX-EX	
lw $s1, 0($t0)							IF	ID	EX	M	WB	
addi $s3, $s1, 10								IF	ID	ID	EX	
sw $s3, 0($t1)									IF	IF	ID	
addi $t0, $t0, 4											IF	
addi $t1, $t1, -4												





Ciclos totales = 6 + 10 * (N- 1) + 13.

Como N = 4(núm it).

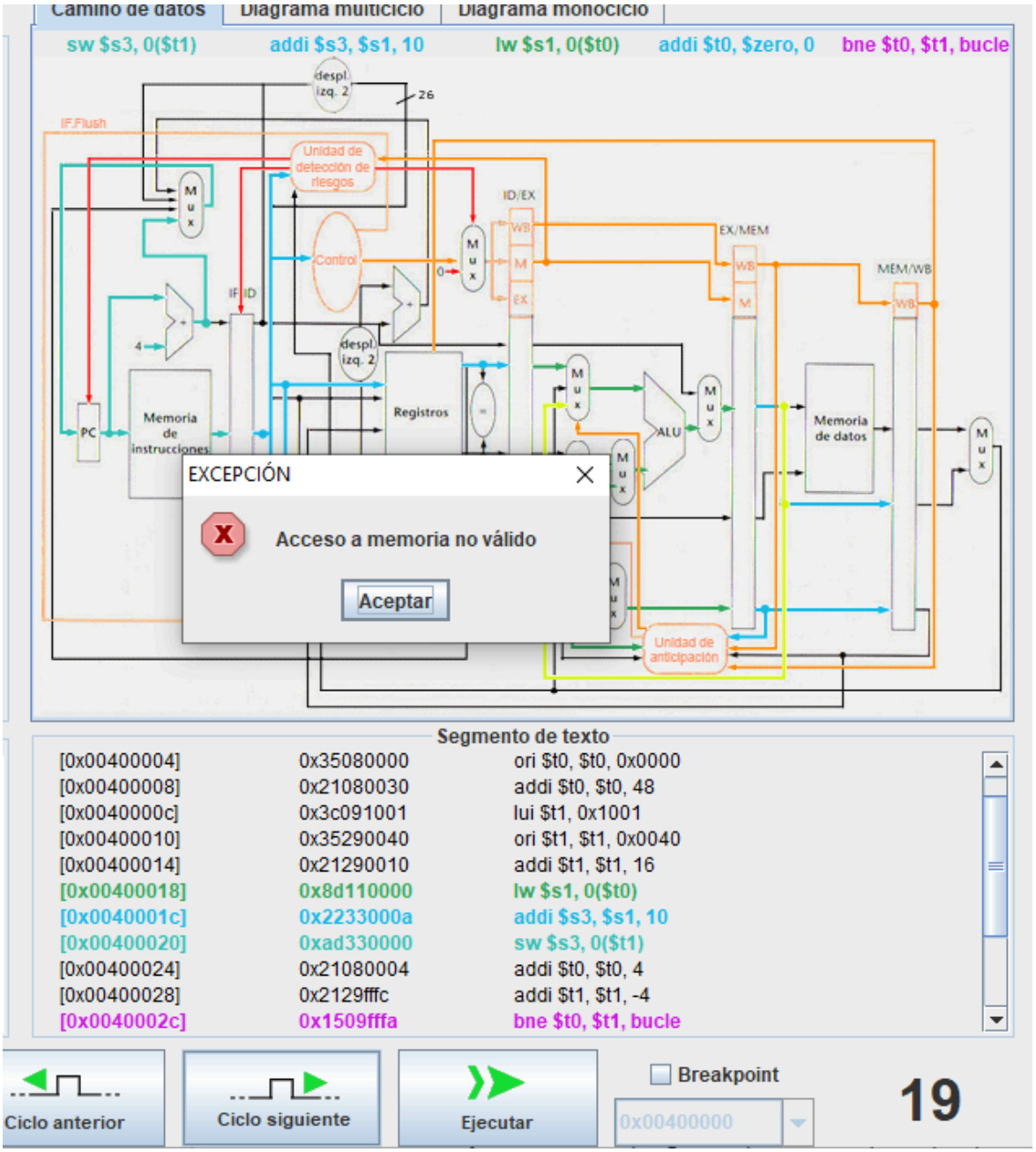
Ciclos totales = 6 + 10*3 + 13 = 49.

Coincide con el resultado del simulador.

Aquí está el cronograma para verlo mejor.
MartinPau_AagaardDaniel_P2-3_Cronograma8

9. Vuelve al editor, ahora cambia la configuración del procesador a *Camino de datos -> Segmentado* pero el salto es *Salto retardado*.

a) ¿Qué ocurre cuando intentas ejecutar? ¿Por qué crees que ocurre? Para darte cuenta de qué pasa ejecuta paso a paso hasta llegar al salto bne, ¿qué ha cambiado respecto a la configuración anterior?



Cuando ejecutamos, el programa peta. Esto pasa ya que estamos accediendo a una posición de memoria no válida debido al salto retardado. Ya que se estará ejecutando la instrucción addi $t0, $zero, 0 que machaca el valor de la dirección de memoria.

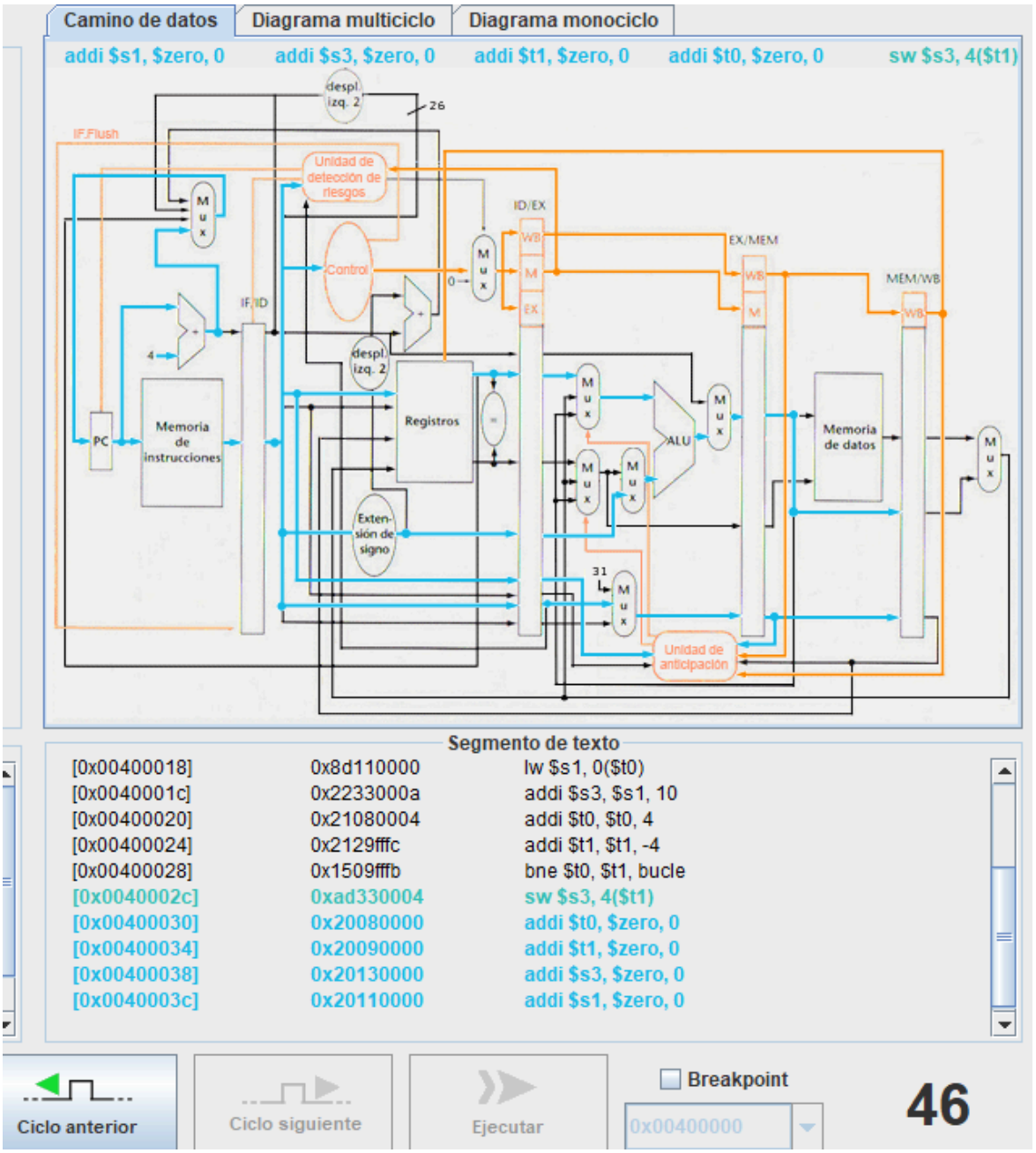
En la configuración anterior esto no pasaba ya que el registro $t0, no se machacaba. Ahora, con el salto retardado, se modifica. Por eso ahora el programa peta y antes no lo hacía.

b) Para solucionar el problema pon una instrucción de dentro del bucle detrás de bne ¿Cuál es la única instrucción posible de poner para que el código haga lo mismo? Si es necesario modifica ligeramente la instrucción movida.

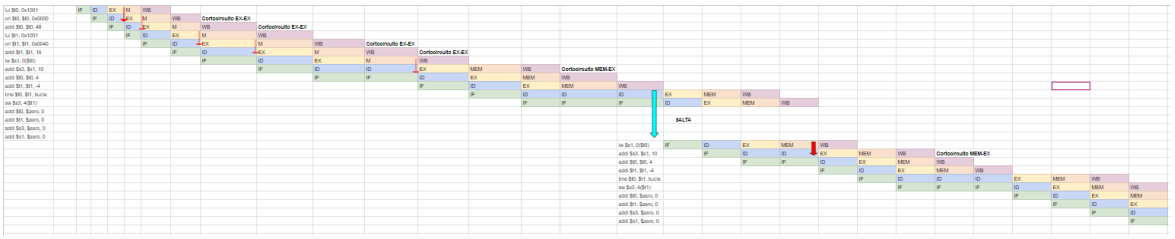
Hemos cambiado la instrucción *sw $s3, 0($t1)* por *sw $s3, 4($t1)* detrás del bne.

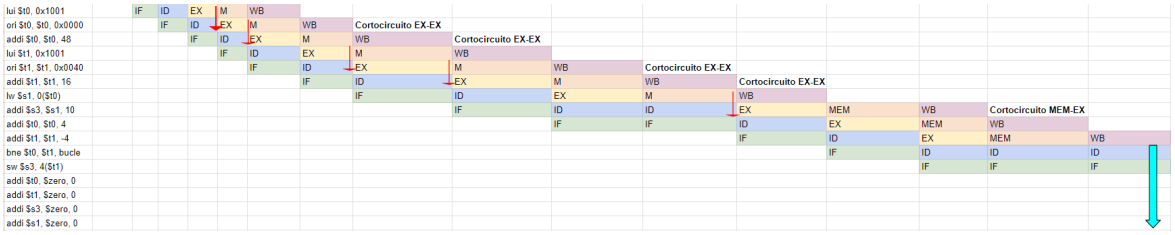
```
.data

X: .word 1, 2, 3, 4, -1, 3, 4, 3, -3, 6, -1, 4 , 0, 5, -1, 0

Y: .space 64

.text

.globl main

        la $t0, X

        addi $t0, $t0, 48

        la $t1, Y

        addi $t1, $t1, 16

bucle:

        lw $s1, 0($t0)

        addi $s3, $s1, 10

        addi $t0, $t0, 4

        addi $t1, $t1, -4

        bne $t0, $t1, bucle

        sw $s3, 4($t1)

        addi $t0, $zero, 0

        addi $t1, $zero, 0

        addi $s3, $zero, 0

        addi $s1, $zero, 0
```

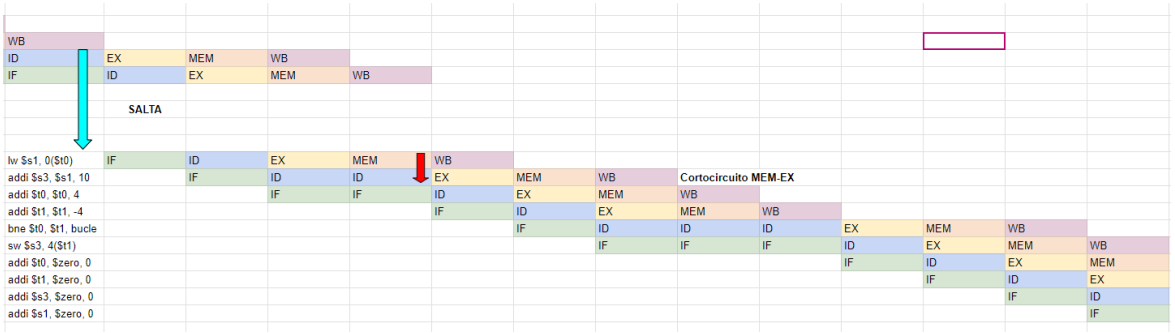
c) Vuelve a ejecutar ahora asegurándote que el resultado es correcto, ¿cuántos ciclos tarda? ¿Tarda menos? Si es así, ¿por qué tarda menos? Haz un cronograma de las dos primeras iteraciones.



Una vez ejecutado el programa de nuevo, tarda 46 ciclos, 3 menos que antes, esto se debe a que hace el salto 3 veces (4 iteraciones) i estamos ahorrando una instrucción por salto de modo que antes de hacer este cambio y con salto fijo hacíamos 49 ciclos y ahora con salto retardado, 46 ciclos.







Para verlo mejor: MartinPau_AagaardDaniel_P2-3_Cronograma9