Examen final

25 de junio de 2025

Fundamentos de computadores II

Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Ejercicios 1 y 2

```
palindrome:
  li    t3, 1
  addi  a1, a1, -1
  slli  a1, a1, 2
  add   a1, a0, a1
do:
  lw    t0, 0(a0)
  lw    t1, 0(a1)
if:
  beq   t0, t1, eif
  li    t3, 0
eif:
  addi  a0, a0, 4
  addi  a1, a1, -4
while:
  blt   a0, a1, do
  mv    a0, t3
  ret
main:
  la sp, _stack
  la a0, array
  li a1, N
  call palindrome
  la t0, lo_es
  sb a0, 0(t0)
  j  .
```

Instrucciones: 20×4B = 80B
Datos: 8×4B + 1B = 33B
Total: 80B + 33B = 113B

no se ejecuta
el array es palíndromo

4 veces

3 salta, 1 no salta

①

N = 4 + [4 + (4×6) + 2] + 2 = **36**

```
palindrome:
  li    t3, 1
  addi  a1, a1, -1
  slli  a1, a1, 2
  add   a1, a0, a1
do:
  lw    t0, 0(a0)
  lw    t1, 0(a1)
if:
  beq   t0, t1, eif
  li    t3, 0
eif:
  addi  a0, a0, 4
  addi  a1, a1, -4
while:
  blt   a0, a1, do
  mv    a0, t3
  sb t3, 0(a2)
  ret
main:
  la sp, _stack
  la a0, array
  li a1, N
  la a2, lo_es
  call palindrome
  la t0, lo_es
  sb a0, 0(t0)
  j  .
```

②

Ejercicio 3

- Las instrucciones de salto condicional utilizan direccionamiento relativo al PC, lo que quiere decir que el valor inmediato codifica con signo el desplazamiento que, sumado al valor actual del PC, indica la dirección de la instrucción a la que saltar.

  La instrucción `beq t0,t1,eif` ocupa la dirección 24
  La etiqueta `eif` señala a la instrucción `addi a0,a0,4` que ocupa la dirección 32
  El valor inmediato será 32-24 = 8 (salto positivo, hacia delante de 2 instrucciones)

  La instrucción `blt a0,a1,do` ocupa la dirección 40
  La etiqueta `do` señala a la instrucción `lw t0,0(a0)` que ocupa la dirección 16
  El valor inmediato será 16-40 = -24 (salto negativo, hacia atrás de 6 instrucciones)

- Al ser saltos relativos al PC, los desplazamientos siempre son los mismos con independencia de donde se ubique la función en memoria.

Ejercicio 4

MemWr = “1” ⇒ estado S5, `sb a0,0(t0)`

ResSrc = “00” y BRWr = “1” ⇒ estado S7, cualquier instrucción `add, addi, jal`

Branch = “1” y Zero = “1” ⇒ estado S10, `beq t0,t1,eif`

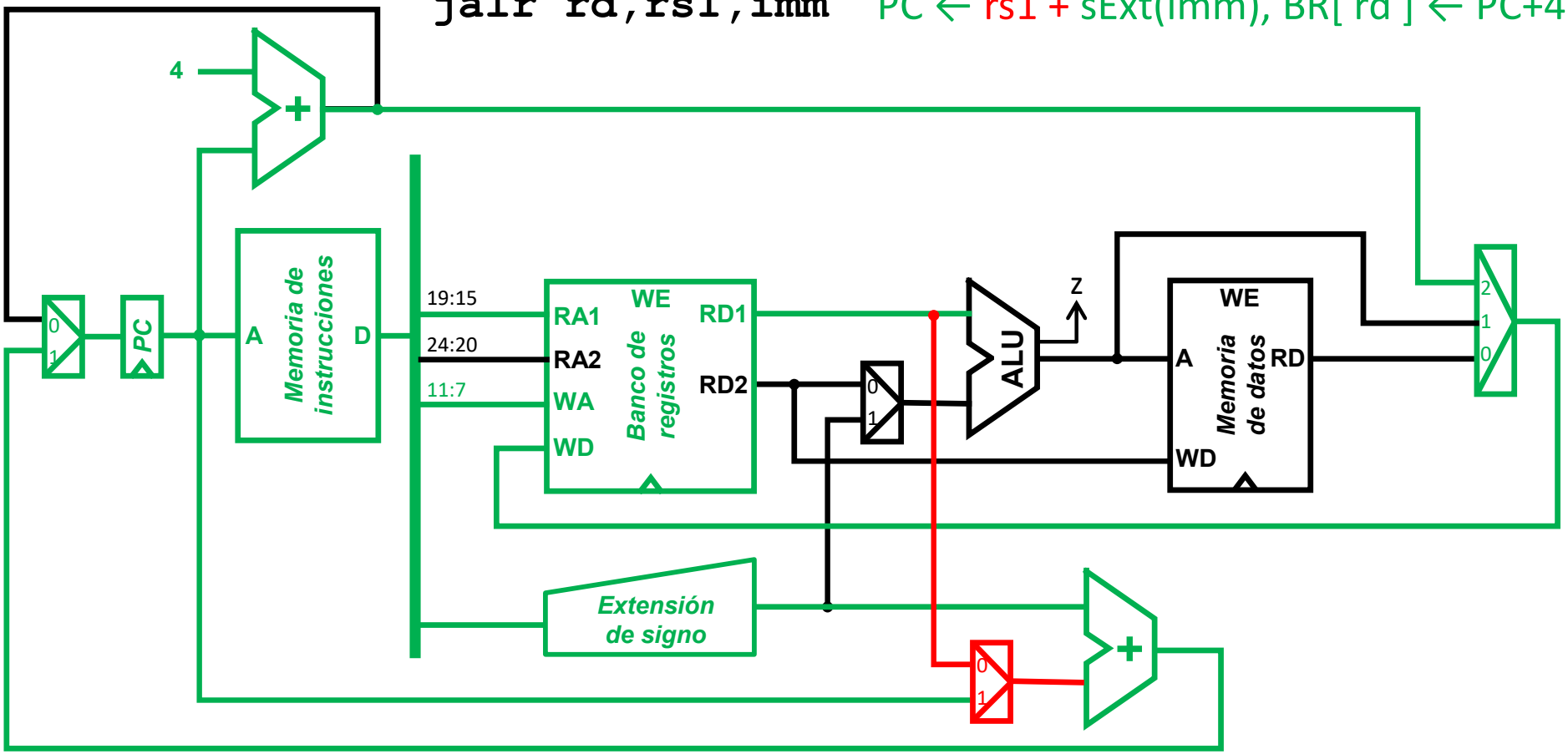
MDRwr = “1” ⇒ estado S3, cualquier instrucción `lw`

Ejercicio 5

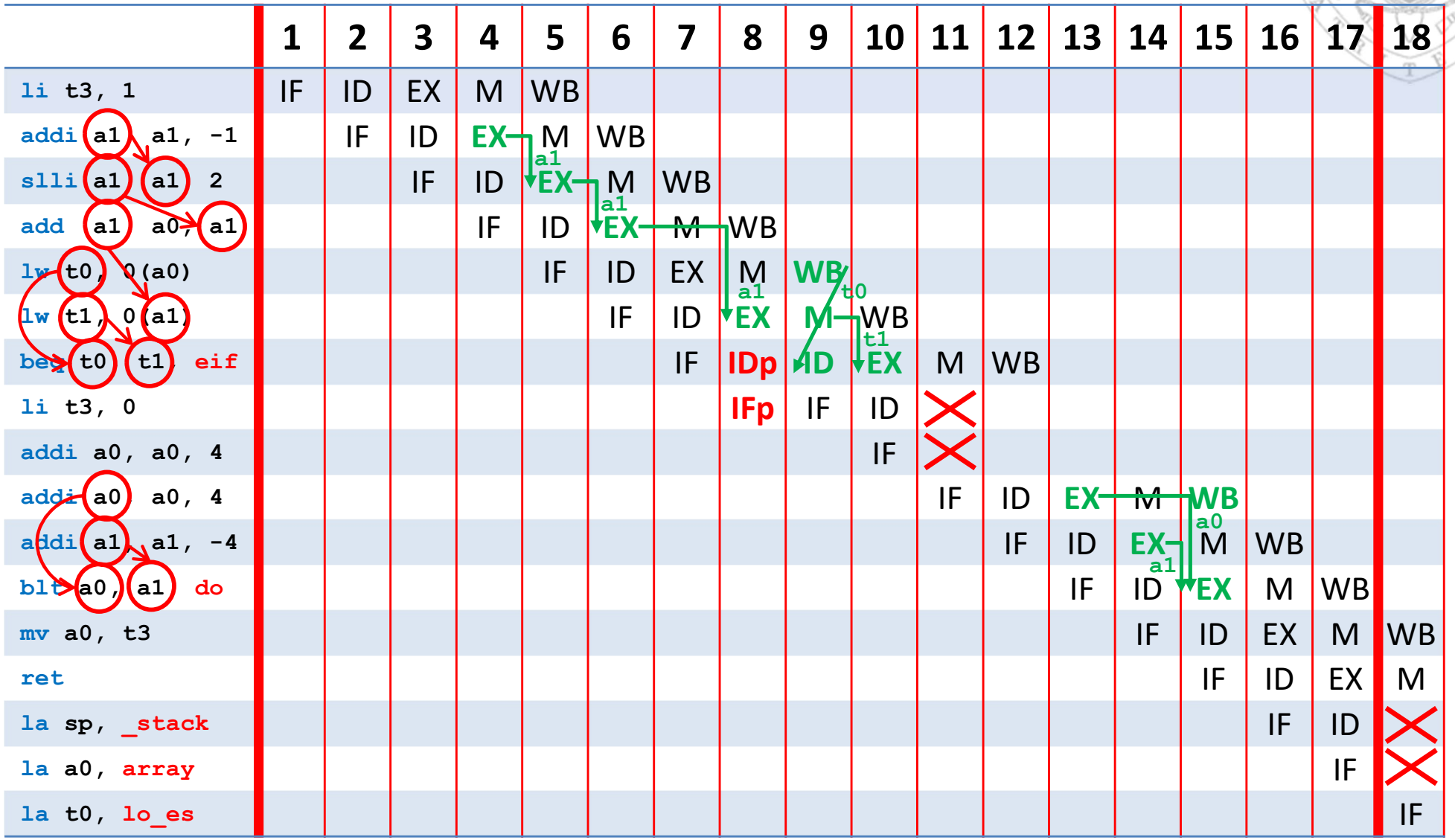
- La mayor parte de la ruta de datos que se usa para la instrucción **`jal`** puede reutilizarse para la instrucción **`jarl`**, solo es necesario un mux adicional.

`jal rd,imm` PC ← PC + sExt(imm), BR[rd] ← PC+4

`jalr rd,rs1,imm` PC ← rs1 + sExt(imm), BR[rd] ← PC+4

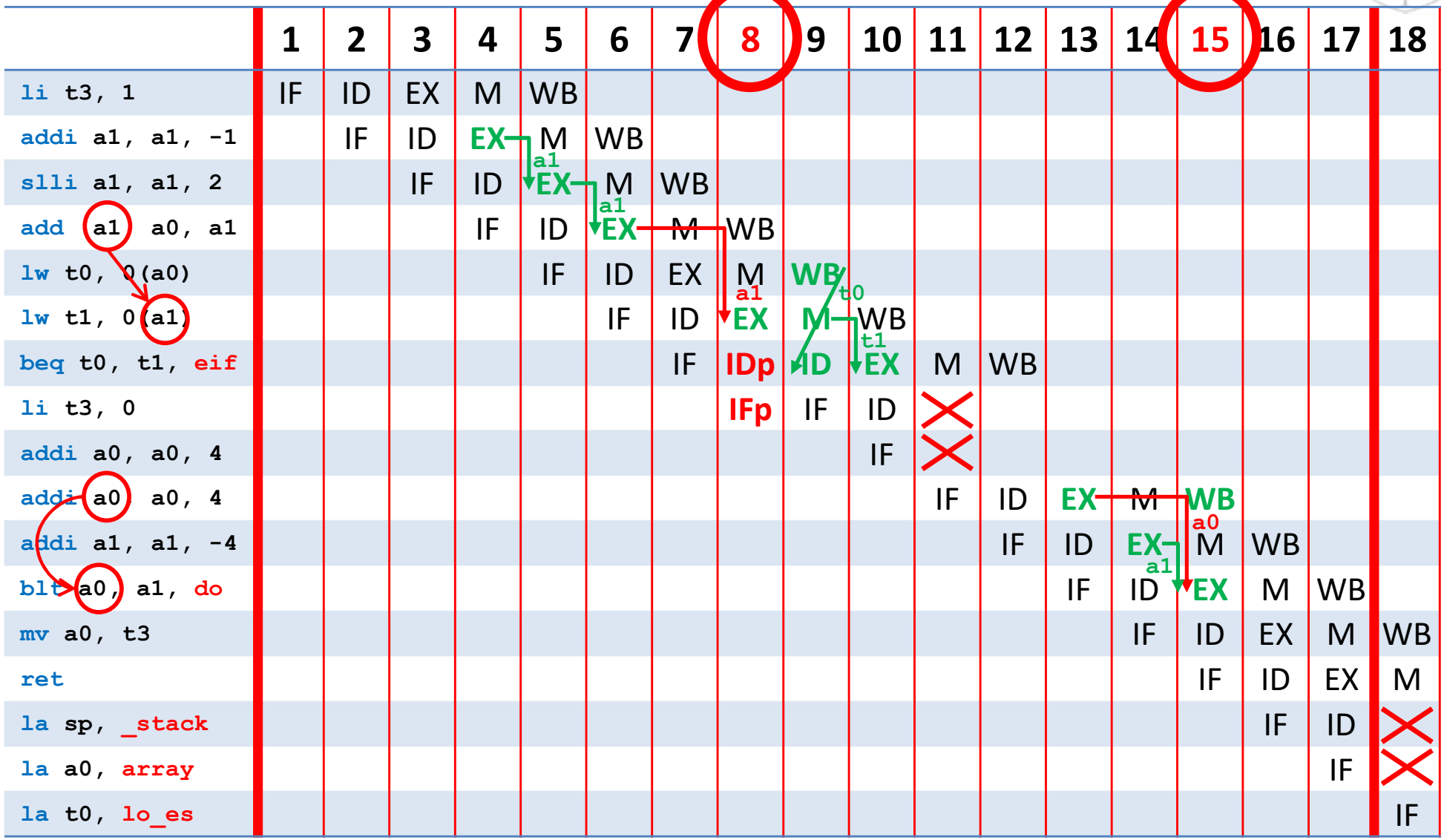


Ejercicio 6


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
li t3, 1
addi a1, a1, -1
slli a1, a1, 2
add a1, a0, a1
lw t0, 0(a0)
lw t1, 0(a1)
beq t0, t1, eif
li t3, 0
addi a0, a0, 4
addi a0, a0, 4
addi a1, a1, -4
blt a0, a1, do
mv a0, t3
ret
la sp, _stack
la a0, array
la t0, lo_es
IF ID EX M WB
EX M WB
EX M WB
EX M WB
IDp IFp
WB
ciclos = 17

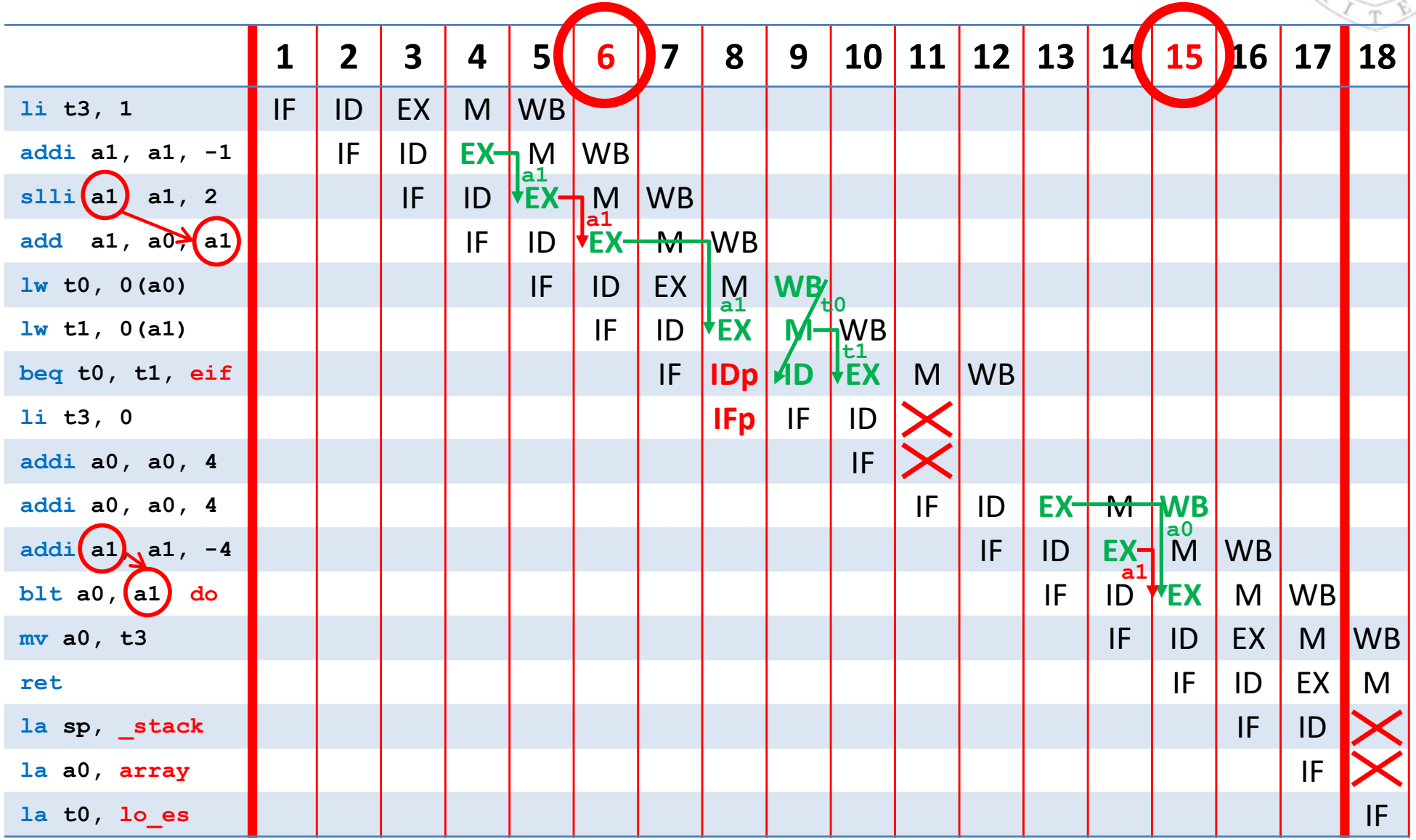
Ejercicio 7a

- ForwardA vale “01” cuando el valor del rs1 se anticipa desde la etapa WB



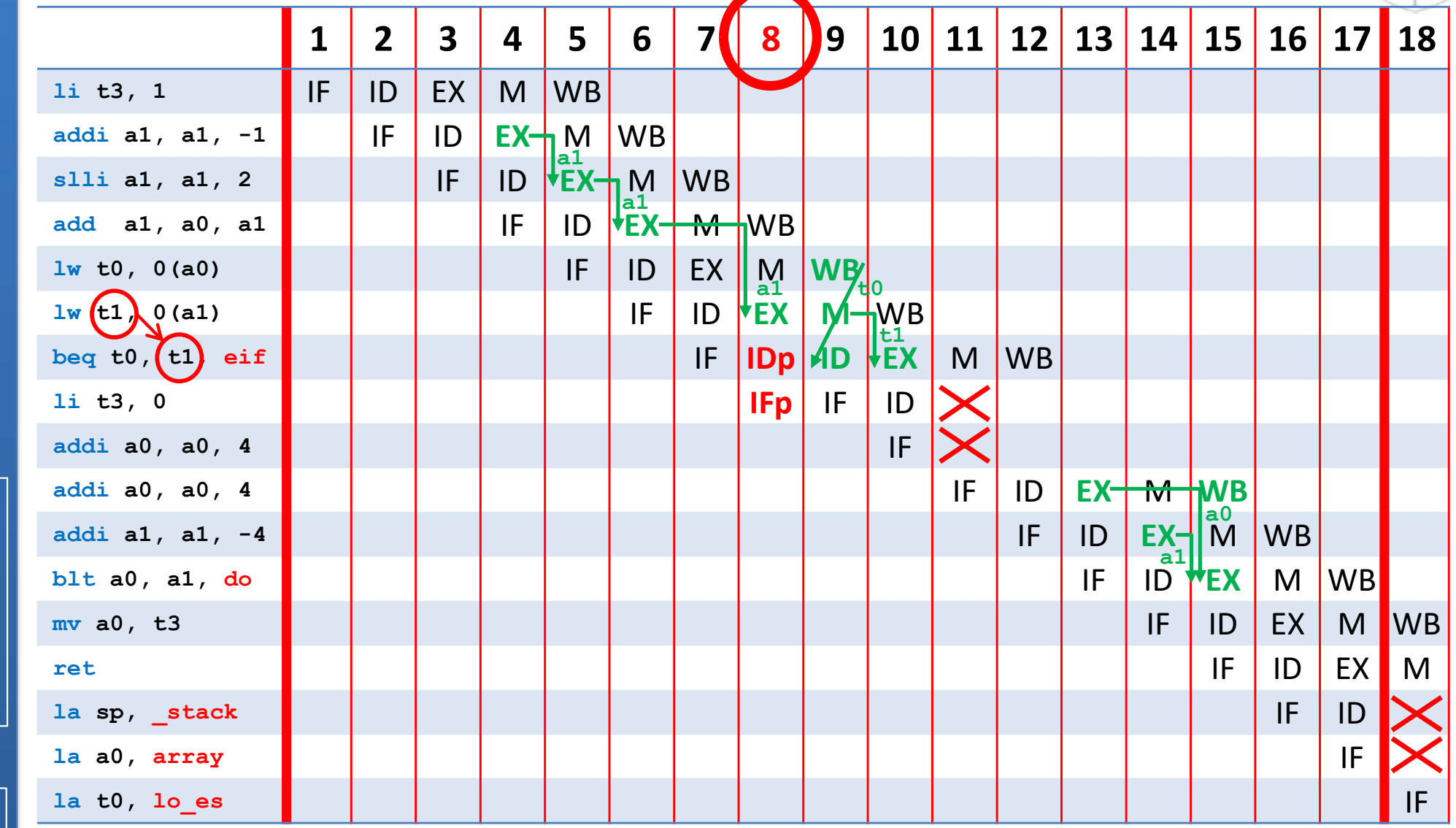
Ejercicio 7b

- ForwardB vale “10” cuando el valor del rs2 se anticipa desde la etapa MEM



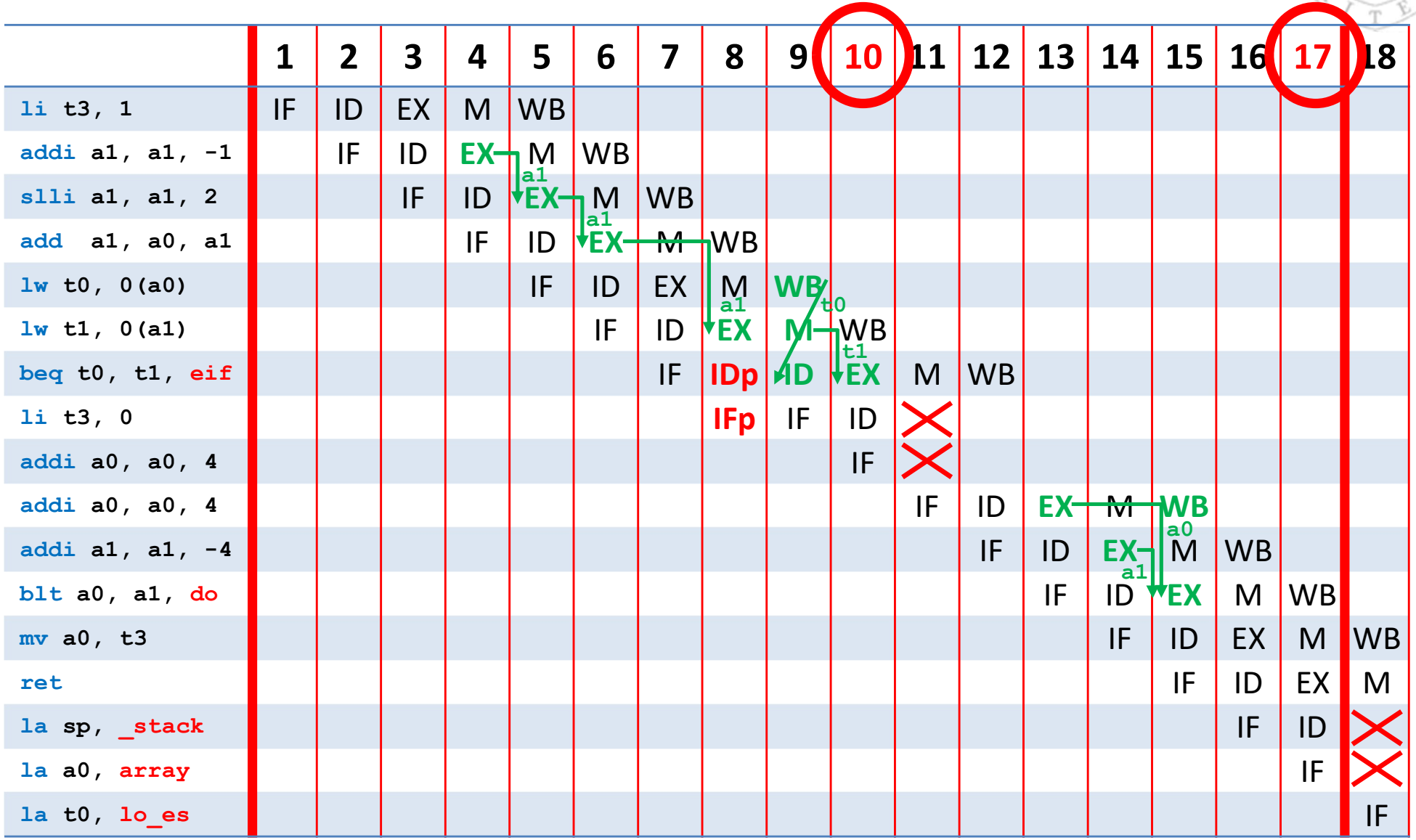
Ejercicio 7c

- StallF y StallD valen ambas “1” cuando hay que parar el pipeline



Ejercicio 7d

- FlushD vale “1” cuando en EX hay una instrucción de salto y ha resuelto saltar



Ejercicio 8

```
palindrome:
  li   t3, 1
  addi a1, a1, -1
  slli a1, a1, 2
  add  a1, a0, a1
do:
  lw   t0, 0(a0)
  lw   t1, 0(a1)
if:
  beq  t0, t1, eif
  li   t3, 0
eif:
  addi a0, a0, 4
  addi a1, a1, -4
while:
  blt  a0, a1, do
  mv   a0, t3
  ret
```

$ciclos_A = 17$

```
palindrome:
  li   t3, 1
  addi a1, a1, -1
  slli a1, a1, 2
  add  a1, a0, a1
do:
  lw   t0, 0(a0)
  lw   t1, 0(a1)
  addi a0, a0, 4
if:
  beq  t0, t1, eif
  li   t3, 0
eif:
  addi a1, a1, -4
while:
  blt  a0, a1, do
  mv   a0, t3
  ret
```

$ciclos_A = 16$

$$t_{ejec} = ciclos \cdot t_{clk}$$

$$Speedup = \frac{t_{ejec}^A}{t_{ejec}^B} = \frac{17}{16} = 1.06$$

Ejercicio 9

```
palindrome:
  li   t3, 1
  addi a1, a1, -1
  slli a1, a1, 2
  add  a1, a0, a1
do:
  lw   t0, 0(a0)
  lw   t1, 0(a1)
if:
  beq  t0, t1, eif
  li   t3, 0
eif:
  addi a0, a0, 4
  addi a1, a1, -4
while:
  blt  a0, a1, do
  mv   a0, t3
  ret
main:
  la sp, _stack
  la a0, array
  li a1, N
  call palindrome
  la t0, lo_es
  sb a0, 0(t0)
  j  .
```

no se ejecuta
el array es palíndromo

200000/2 = 10000 veces

20000/2-1 veces =
9999 salta, 1 no salta

	monociclo	multiciclo	segmentado
	1	4	1
	1	4	1
	1	4	1
	1	4	1
	1	5	1
	1	5	1 + 1
	1	3	3
	0	0	0
	1	4	1
	1	4	1
	1	3	1 o 3
	1	4	1
	1	4	3
	1	4	1
	1	4	1
	1	4	1
	1	4	3
	1	4	1
	1	4	1

N = 4 + [4 + (10000×6) + 2] + 2 = **60012**

Ejercicios 9 y 10

⑨

	N	ciclos	CPI	f_{clk}	t_{ciclo}	t_{ejec}	MIPS
monociclo	60012	60012	1	500 MHz	2 ns	120 µs	500
multiciclo	60012	240048	4	1 GHz	1 ns	240 µs	250
segmentado	60012	110014	1.83	2 GHz	0.5 ns	55 µs	1093

⑩

a. Excepción de dirección de dato en lectura desalineada.
b. La provocará la ejecución de la instrucción `lw t1,0(a1)` que generará una dirección impar. La instrucción está en la dirección 0x14 luego mepc = 0x14. La dirección de lectura 0x10000+(8-1) luego mtval = 0x10007
c. En el ciclo 14, ya que la instrucción comienza a ejecutarse en el ciclo 11 y este tipo de excepción se dispara en la etapa MEM.
d. En el ciclo 15.

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)
  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/