Facultad de Informática
Universidad Complutense de Madrid

EXAMEN FINAL – FUNDAMENTOS DE COMPUTADORES II

25 DE JUNIO 2025

Se dice que un array es palíndromo si sus componentes están dispuestos de manera que resulta el mismo leído de izquierda a derecha que de derecha a izquierda. Es fácil comprobarlo si se comparan sucesivamente dos a dos los elementos que están en la misma posición relativa respecto a los extremos del array.

Sea el siguiente programa en ensamblador del RISC-V que efectúa el algoritmo descrito. Suponga en todos los ejercicios que, durante la fase de ensamblado, cada pseudointrucción se traduce a una única instrucción (la indicada como comentario), que la sección **`.text`** se ubica a partir de la dirección 0x0 de memoria y que la secciones **`.data`**, **`.bss`** se ubican consecutivamente a partir de la dirección 0x10000.

```
.global main
.equ  N, 8
.data
  array: .word 3, 1, 4, 1, 1, 4, 1, 3
.bss
  lo_es: .space 1
.text
palindrome:
  li   t3, 1            // addi t3, x0, 1
  addi a1, a1, -1       // calcula dir del ultimo elemento del array
  slli a1, a1, 2
  add  a1, a0, a1
do:
  lw   t0, 0(a0)        // a0 apunta al primer elemento del array
  lw   t1, 0(a1)        // a1 apunta al ultimo elemento del array
if:
  beq  t0, t1, eif      // compara elementos
  li   t3, 0            // addi t3, x0, 0
eif:
  addi a0, a0, 4        // mueve a0 a la derecha
  addi a1, a1, -4       // mueve a1 a la izquierda
while:
  blt  a0, a1, do       // chequea si los punteros se han alcanzado
  mv   a0, t3           // addi a0, t3, 0
  ret                   // jalr x0, ra, 0
main:
  la   sp, _stack       // addi sp, x0, _stack
  la   a0, array        // addi a0, x0, array
  li   a1, N            // addi a1, x0, N
  call palindrome
  la   t0, lo_es        // addi t0, x0, lo_es
  sb   a0, 0(t0)
  j    .                // jal  x0, .
.end
```

1. **(0,25 puntos)**. Indique el tamaño en bytes que ocupa el programa completo contando instrucciones y datos. Calcule el número de instrucciones (sin contar `j .`) que se ejecutan al correr este programa.

2. **(0,5 puntos)**. La función **`palindrome`** devuelve 1 ó 0 dependiendo de si el array que se le pasa como argumento es palíndromo o no. La función **`main`** almacena dicho valor en la dirección de memoria etiquetada como **`lo_es`**. Recodifique el programa para que tenga idéntico comportamiento, pero sea la propia función **`palindrome`** la encargada de almacenar 1 ó 0 en la dirección que la función **`main`** le indique.

3. **(0,75 puntos)**. El formato de las instrucciones de salto condicional incluye un campo de desplazamiento para indicar un valor inmediato de 13 bits. Indique el valor en decimal de dicho desplazamiento en las instrucciones de salto condicional existentes en la función **`palíndrome`**. ¿Cambiaría dicho valor si la sección **`.text`** se ubicara en otra posición? Justifíquelo.

4. **(1 punto)**. Suponiendo que el programa se ejecuta en el procesador multiciclo y conociendo el valor que en cierto ciclo toman en cada caso algunas de las señales de este, indique el estado en que se encuentra en dicho ciclo (si pueden ser varios indíquelos todos) y la instrucción que está ejecutando (si son varias las posibles, indique una de ellas):
   a) MemWr = "1"
   b) ResSrc = "00" y BRWr = "1"
   c) Branch = "1" y Zero = "1"
   d) MDRwr ="1"

5. **(1 punto)**. Por un lado, sabemos que durante la fase de ensamblado la pseudoinstrucción **`ret`** se traduce **`jalr x0,ra,0`**. Por otro lado, también sabemos que la funcionalidad de **`jalr rd,rs1,imm`**$_{12b}$ es: **`PC ⟵ rs1+sExt(imm), rd ⟵ PC+4`**. Discuta las modificaciones, si es que fueran necesarias, a realizar en la ruta de datos del procesador monociclo de repertorio reducido para que pueda ejecutar instrucciones **`jalr`**.

6. **(2 puntos)**. Suponiendo que N = 1, realice el diagrama de ejecución de la función **`palindrome`** en el procesador segmentado*. Deberá mostrarse desde el ciclo en que se lanza la primera instrucción de la función hasta el ciclo en que se lanza la primera instrucción del programa principal tras retornar de la función. Suponga que todas las instrucciones de salto tienen un comportamiento análogo.

7. **(1 punto)**. A la vista del diagrama de ejecución del ejercicio 6 indique:
   a) Los ciclos en que la señal ForwardA vale "01"
   b) Los ciclos en que la señal ForwardB vale "10"
   c) Los ciclos en que la señales StallF y StallD valen ambas "1".
   d) Los ciclos en que la señal FlushD vale "1".

8. **(0.5 puntos)**. Reordene las instrucciones de la función **`palindrome`** para reducir la penalización por paradas en el procesador segmentado*. Calcule el *speedup* resultante de la optimización para N=1.

9. **(2 puntos).** Considerando que el programa dado trabajase con un array palíndromo de 20000 palabras, complete la siguiente tabla*. Suponga que todas las instrucciones de salto tienen un comportamiento análogo en las microcroarquitecturas monociclo y

* El procesador segmentado tiene gestión completa de riesgos (escritura del BR a mitad de ciclo, unidad de anticipación y unidad de conflictos con predicción de salto no tomado).

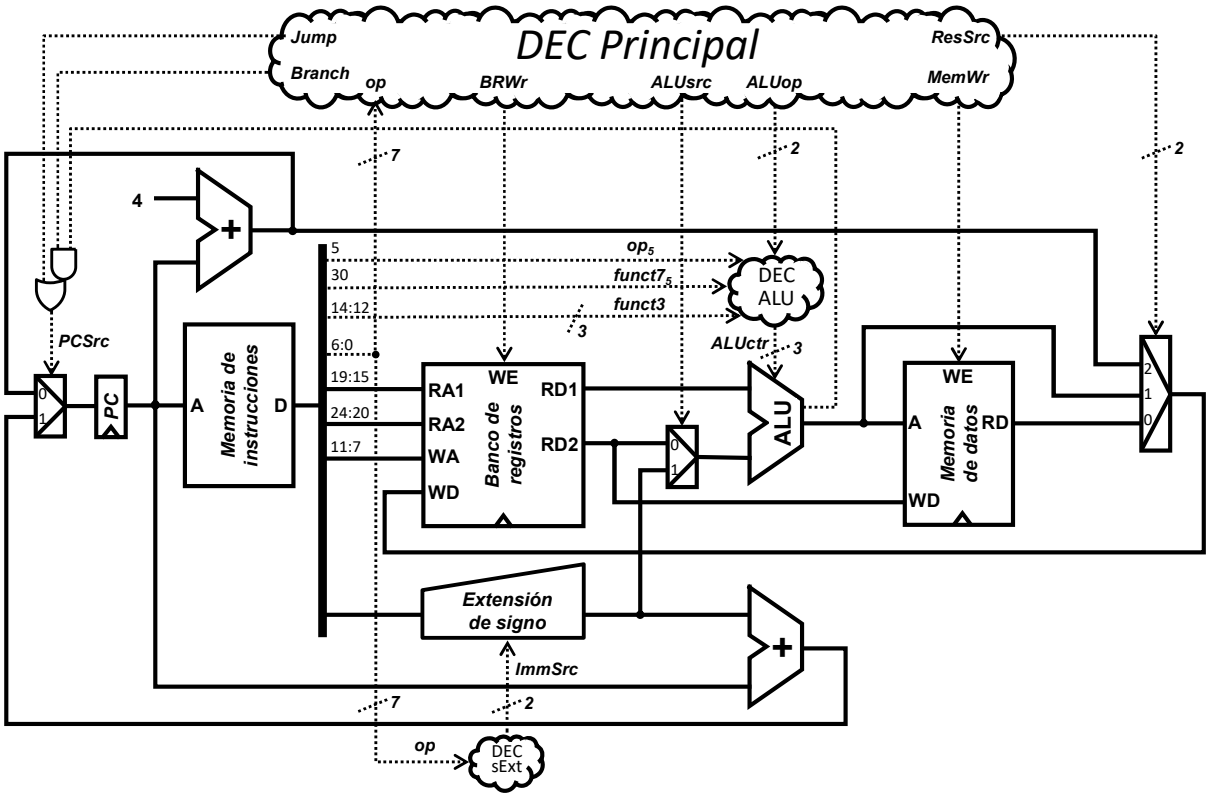
segmentada y que, en el caso del procesador multiciclo, la instrucción **`jalr`** tarda 4 ciclos.

	núm. instrucciones ejecutadas	ciclos	CPI	f_{clk}	t_{ejec}	MIPS
Monociclo						500
Multiciclo				1 GHz		
Segmentado					55 µs	

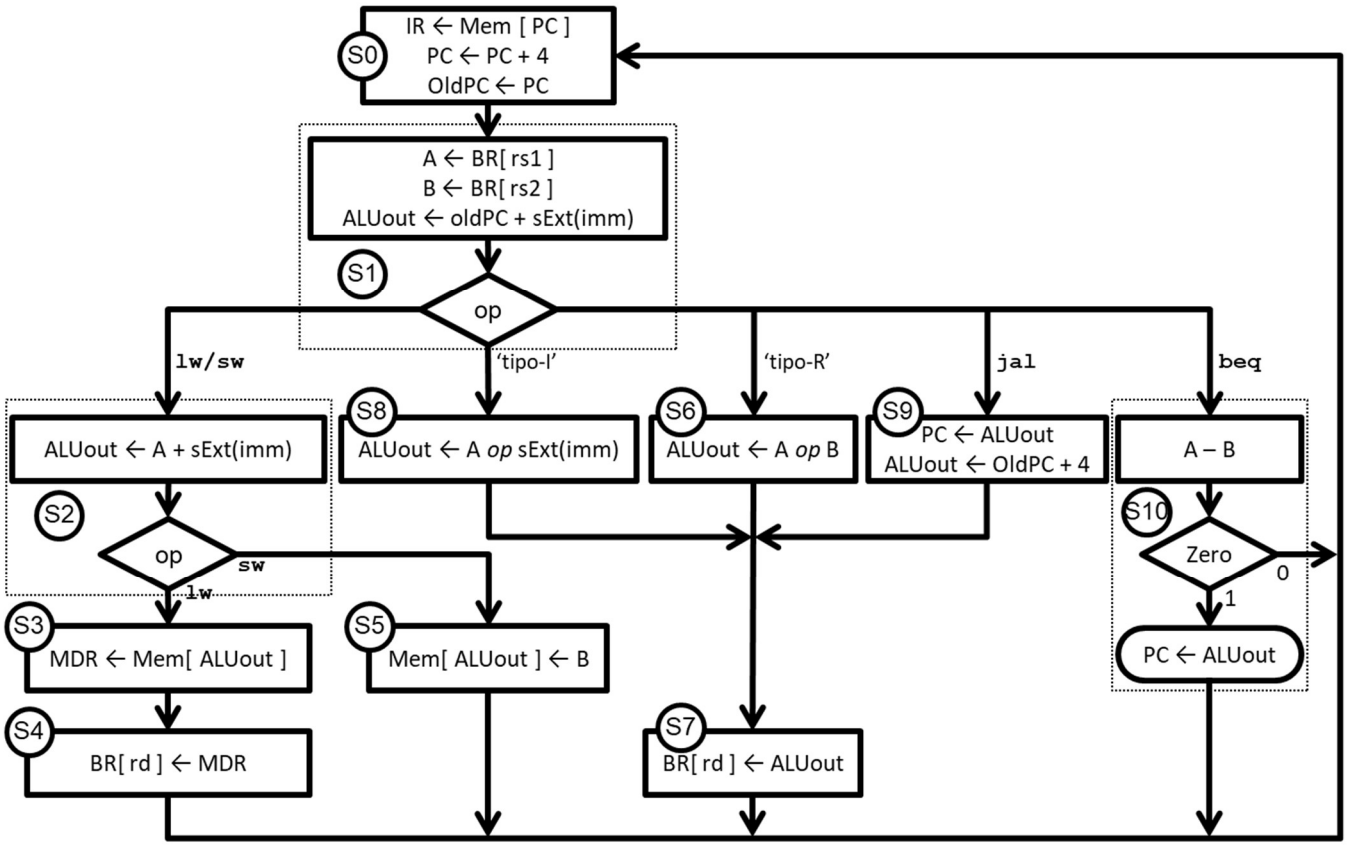
10. (1 punto). Suponga que un programador, por error, olvida escribir la instrucción **`slli a1,a1,2`** en la función **`palindrome`**. Al ejecutar el programa erróneo en el procesador RISC-V de microarquitectura segmentada se disparará una excepción, indique:

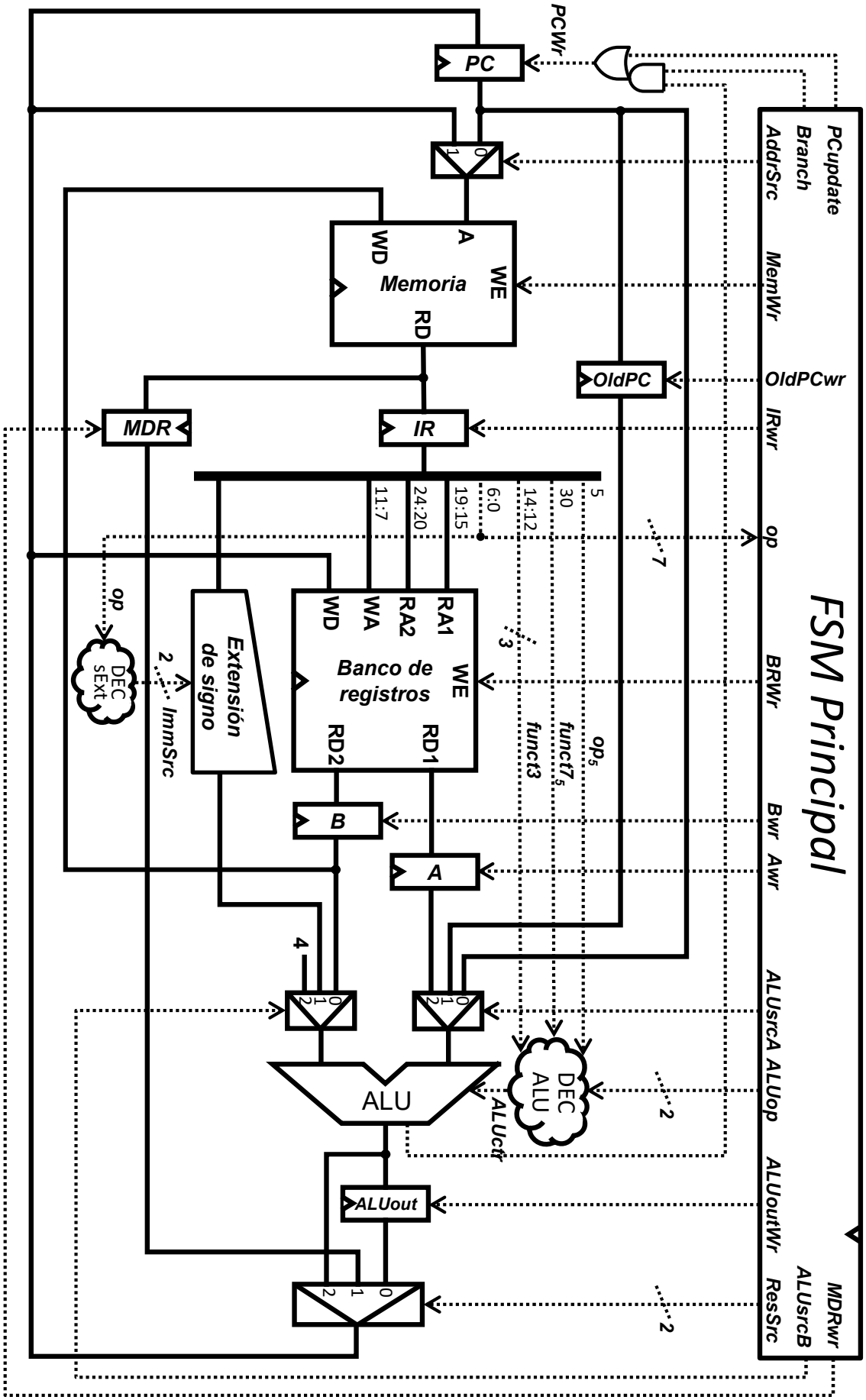
a) Tipo de excepción y justificación de por qué se produce.
b) Valores que se almacenarán en los registros **`mepc`** y **`mtval`**.
c) Ciclo en que se disparará la excepción.
d) Ciclo en que se lanzará la primera instrucción de la rutina de tratamiento de excepción.

MICROARQUITECTURA MONOCICLO



MICROARQUITECTURA MULTICICLO




PCWr
PC
PCupdate
Branch
AddrSrc
MemWr
OldPCwr
IRwr
op
BRWr
Bwr
Awr
ALUsrcA
ALUop
ALUoutWr
ALUsrcB
ResSrc
MDRwr
FSM Principal
Memoria
WD
A
WE
RD
OldPC
MDR
IR
11:7
24:20
19:15
6:0
14:12
30
5
7
3
op
DEC
sExt
2
ImmSrc
Extensión
de signo
Banco de
registros
WA
RA2
RA1
RD2
RD1
op5
funct7
funct3
B
A
4
ALU
DEC
ALU
ALUctr
ALUout
2
2

MICROARQUITECTURA SEGMENTADA