Facultad de Informática
Universidad Complutense de Madrid

# EXAMEN FINAL – FUNDAMENTOS DE COMPUTADORES II

24 DE MAYO 2024

Sea el siguiente programa en ensamblador del RISC-V y suponga en todos los ejercicios que las pseudointrucciones **`la`** y **`li`** se traducen a una única instrucción **`addi`** en la fase de ensamblado, 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, 6
.data
  f: .word  2
.bss
  Res: .space 4
.text
main:
  la  t1, f              # t1 = &f
  lw  s1, 0(t1)          # s1 = f
  li  s2, n              # s2 = n
  li  s3, 2              # s3 = 2 (índice)
for:
  beq  s3, s2, fin_for
  add  s1, s1, s1
  addi s3, s3, 1
  jal  x0, for           # j for
fin_for:
  la t1, Res
  sw s1, 0(t1)
end:
  j  .
.end
```

1. **Monociclo [0,5 pt]**. En el procesador monociclo indique los valores de los registros y posiciones de memoria mostrados a continuación en cada uno de los ciclos de la primera iteración del bucle.

| Num. ciclo | Instrucción | PC | s1 | s2 | s3 | M[0x10000] |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| ... | | | | | | |

2. **Monociclo [0,5 pt]**. En el procesador monociclo indique los valores de las señales de control mostradas a continuación en cada uno de los ciclos de la primera iteración del bucle.

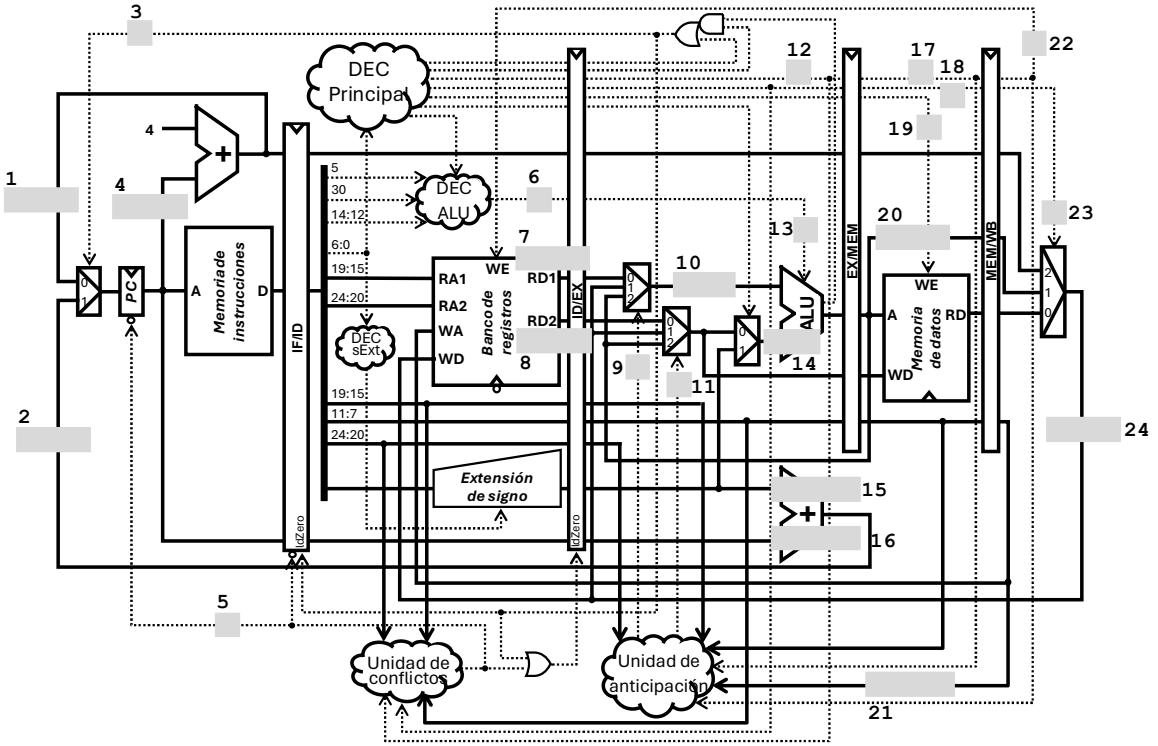
| Num. ciclo | Instrucción | PCsrc | Branch | Jump | BRwr | ALUsrc | MemWr | ResSrc | ImmSrc |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | | | | | | | | | |
| ... | | | | | | | | | |

3. **Multiciclo [1 pt]**. Suponiendo que el programa (hasta la instrucción **`j .`** no incluida) tarda en ejecutarse 58,7 ns en un procesador multiciclo con una frecuencia de reloj de 1,5 GHz.
   a) Calcule el CPI del programa.
   b) Calcule el valor de la métrica MIPS.

4. **Segmentado [1,5 pt]**. En el procesador segmentado sin gestión de conflictos (escritura del BR a final de ciclo):
   a) Inserte en el programa las instrucciones **`nop`** necesarias para que se ejecute correctamente.
   b) Realice el diagrama de ejecución de la primera iteración del bucle.
   c) Calcule los ciclos promedio por instrucción (CPI) en la ejecución completa del bucle*.

5. **Segmentado [1,5 pt]**. En el procesador segmentado con gestión de conflictos completa (escritura del BR a mitad de ciclo, unidad de anticipación y unidad de conflictos con predicción de salto no tomado):
   a) Realice el diagrama de ejecución de las 5 primeras instrucciones.
   b) Calcule el número de ciclos que tarda en ejecutarlas*.
   c) Calcule los ciclos promedio por instrucción (CPI) en la ejecución del programa completo (hasta el lanzamiento de la instrucción `j .` no incluida)*.

6. **Segmentado [2 pt]**. En el procesador segmentado con gestión de conflictos completa mostrado a continuación, indique la instrucción que se encuentra en cada etapa del procesador y los valores que toman las 24 señales indicadas en el ciclo 7 de la ejecución del programa. La correspondencia entre alias y número de registro es: t1 = x6, s1= x9, s2 = x18 y s3 = x19.



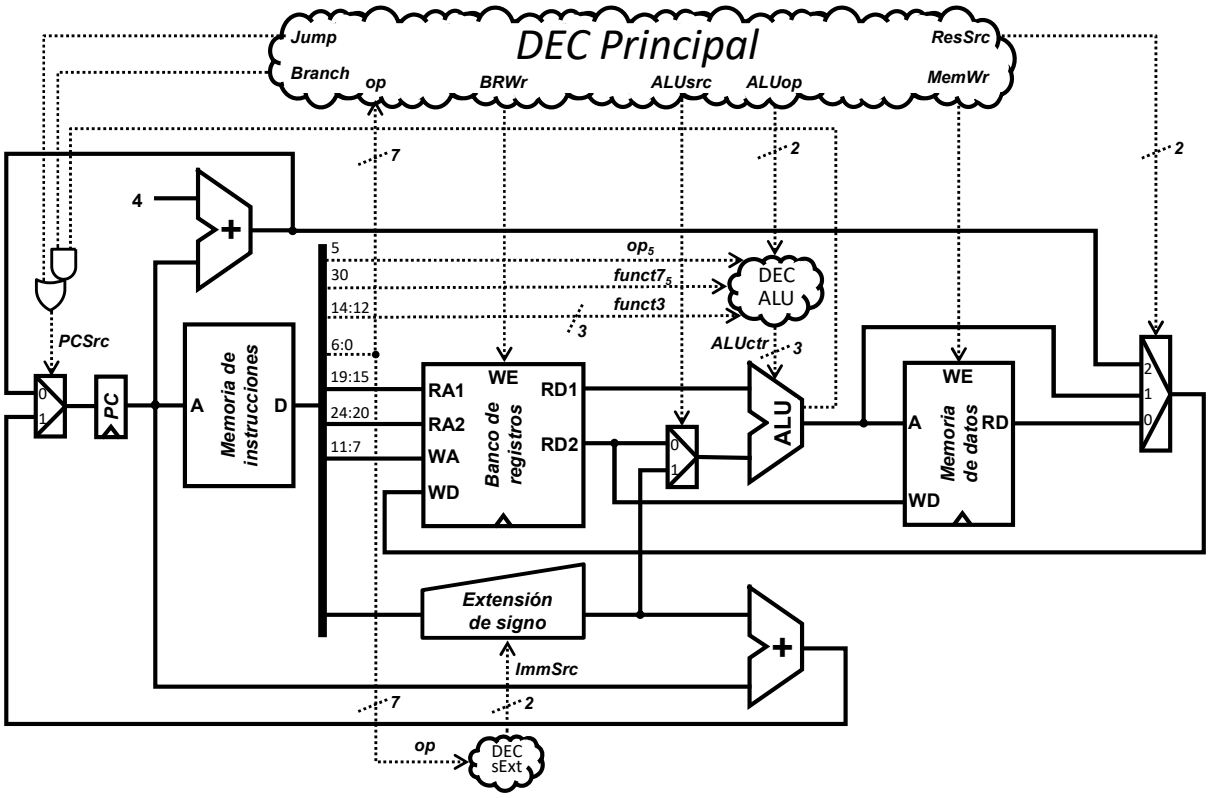

7. **Excepciones [1 pt]**. Suponga que al ejecutar el programa la instrucción **`lw s1,0(t1)`** provoca una excepción en el procesador segmentado, indique:
   a) ¿Qué tipo de excepción sería?
   b) ¿En qué etapa del procesador se produciría?
   c) ¿En qué ciclo se produciría?
   d) ¿Qué circunstancia específica se ha dado para que se produzca esta excepción y por qué?
   e) ¿Qué instrucciones lanzadas se cancelarían?

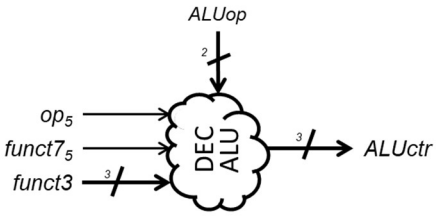
* A la hora de contar los ciclos que tarda en ejecutarse un bloque de código, debe contarse desde el ciclo en que lanza la primera instrucción del bloque (incluido) hasta el ciclo que se lanza la siguiente instrucción al bloque (no incluido).

8. **Ensamblador [1,5 pt]**. Complete el código para que, justo antes de la etiqueta **`end`**, el programa llame a una subrutina de nombre **`subr`** cumpliendo las siguientes condiciones:
   - El programa invocante quiere conservar el valor de **`t1`**.
   - Los valores que deben pasarse como parámetros se encuentran en los registros **`s2`** y **`s3`**.
   - El resultado debe almacenarse en la posición definida por la etiqueta **`res`**.
   - Además, el programa a su inicio debe implementar la inicialización de la pila suponiendo que está ubicada a partir de una dirección indicada por la etiqueta externa **`_stack`**.

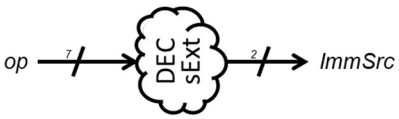
9. **Ensamblador [0,5 pt]**. Codifique el prólogo y el epílogo de la subrutina indicada en el anterior ejercicio suponiendo que es de tipo no-hoja, que utiliza los registros **`s3`** y **`s5`**, y que el resultado final que debe devolver a la función invocante está almacenado en **`s5`**.

# PROCESADOR MONOCICLO

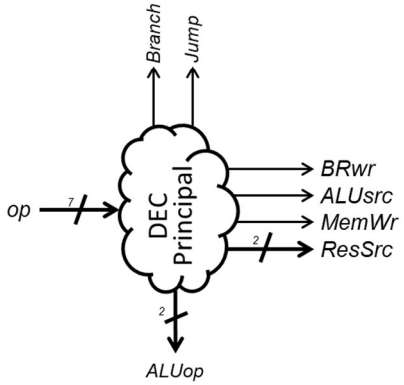



| ALUop | $op_5$ | $funct7_5$ | funct3 | ALUctr |
|---|---|---|---|---|
| $00^{(sumar)}$ | X | X | XXX | $000^{(A+B)}$ |
| $01^{(restar)}$ | X | X | XXX | $001^{(A-B)}$ |
| $10^{(operar)}$ | 0 | X | $000^{(addi)}$ | $000^{(A+B)}$ |
| $10^{(operar)}$ | 1 | 0 | $000^{(add)}$ | $000^{(A+B)}$ |
| $10^{(operar)}$ | 1 | 1 | $000^{(sub)}$ | $001^{(A-B)}$ |
| $10^{(operar)}$ | X | X | $010^{(slt/slti)}$ | $101^{(A<B)}$ |
| $10^{(operar)}$ | X | X | $110^{(or/ori)}$ | $011^{(A\mid B)}$ |
| $10^{(operar)}$ | X | X | $111^{(and/andi)}$ | $010^{(A\&B)}$ |

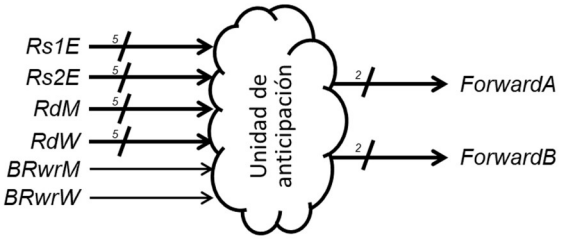



| Op | ImmSrc |
|---|---|
| $0000011^{(lw)}$ | $00^{(tipo-I)}$ |
| $0100011^{(sw)}$ | $01^{(tipo-S)}$ |
| $0010011^{(tipo-I)}$ | $00^{(tipo-I)}$ |
| $0110011^{(tipo-R)}$ | – |
| $1100011^{(beq)}$ | $10^{(tipo-B)}$ |
| $1101111^{(jal)}$ | $11^{(tipo-J)}$ |



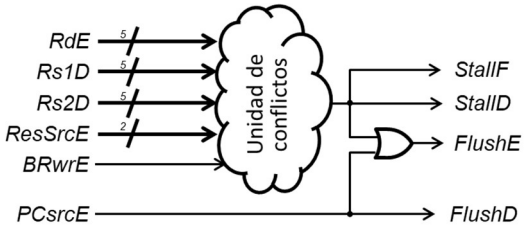

| op | Branch | Jump | BRwr | ALUsrc | ALUop | MemWr | ResSrc |
|---|---|---|---|---|---|---|---|
| $0000011^{(lw)}$ | 0 | 0 | 1 | 1 | $00^{(sumar)}$ | 0 | 00 |
| $0100011^{(sw)}$ | 0 | 0 | 0 | 1 | $00^{(sumar)}$ | 1 | – |
| $0010011^{(tipo-I)}$ | 0 | 0 | 1 | 1 | $10^{(operar)}$ | 0 | 01 |
| $0110011^{(tipo-R)}$ | 0 | 0 | 1 | 0 | $10^{(operar)}$ | 0 | 01 |
| $1100011^{(beq)}$ | 1 | 0 | 0 | 0 | $01^{(restar)}$ | 0 | – |
| $1101111^{(jal)}$ | 0 | 1 | 1 | – | – | 0 | 10 |

# PROCESADOR SEGMENTADO



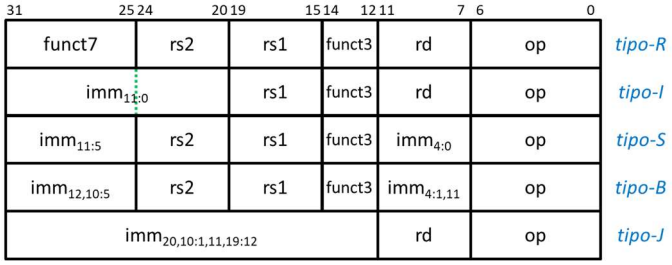

*if* ( (Rs1E ≠ 0) & BRwrM & (Rs1E = RdM) ) *then* ( ForwardA ← $10^{\text{(anticipar MEM)}}$ )
*elsif*( (Rs1E ≠ 0) & BRwrW & (Rs1E = RdW) ) *then* ( ForwardA ← $01^{\text{(anticipar WB)}}$ )
*else* ( ForwardA ← $00^{\text{(no anticipar)}}$ )

*if* ( (Rs2E ≠ 0) & BRwrM & (Rs2E = RdM) ) *then* ( ForwardB ← $10^{\text{(anticipar MEM)}}$ )
*elsif*( (Rs2E ≠ 0) & BRwrW & (Rs2E = RdW) ) *then* ( ForwardB ← $01^{\text{(anticipar WB)}}$ )
*else* ( ForwardB ← $00^{\text{(no anticipar)}}$ )




*if* ( (ResSrcE = 0) & BRwrE & ((Rs1D = RdE) | (Rs2D = RdE)) ) *then* ( lwStall ← 1 ) (parar pipeline)
*else* ( lwStall ← 0 ) (no parar pipeline)
StallF = StallD = lwStall
FlushD = PCsrcE
FlushE = lwStall | PCsrcE

# FORMATO DE INSTRUCCIÓN




| Instrucción | Tipo | **funct7** bits 31:25 | **funct3** bits 14:12 | **op** bits 6:0 |
|---|---|---|---|---|
| **lw** | I | – | 010 | 0000011 |
| **sw** | S | – | 010 | 0100011 |
| **add** | R | 0000000 | 000 | 0110011 |
| **sub** | R | 0100000 | 000 | |
| **slt** | R | 0000000 | 010 | |
| **or** | R | 0000000 | 110 | |
| **and** | R | 0000000 | 111 | |
| **addi** | I | – | 000 | 0010011 |
| **slti** | I | – | 010 | |
| **ori** | I | – | 110 | |
| **andi** | I | – | 111 | |
| **beq** | B | – | 000 | 1100011 |
| **jal** | J | – | – | 1101111 |