Facultad de Informática
Universidad Complutense de Madrid

EXAMEN FINAL – FUNDAMENTOS DE COMPUTADORES II

23 DE MAYO 2025

La calculadora de sobremesa Friden EC-132 (expuesta en el museo de nuestra Facultad) comenzó a comercializarse en 1965 y fue la primera calculadora electrónica capaz de realizar raíces cuadradas. Para calcular esta operación utilizaba una variación del algoritmo iterativo de "Suma de impares". Este algoritmo se basa en el hecho de que el cuadrado de un número N es igual a la suma de los N números impares a partir de 1 ($5^2 = 25 = 1 + 3 + 5 + 7 + 9$). Según esto, restando sucesivamente a un número los impares empezando por el 1, es posible alcanzar la parte entera de su raíz cuadrada como la cantidad de impares restados.

```
int sqrt( int num )
{
  int raiz = 0;
  for( int i = 1; num >= i; i = i+2 )
  {
    num  = num - i;
    raiz = raiz + 1;
  }
  return raiz;
}
```

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).

```
.global main
.extern _stack
.data
  num: .word 25
.bss
  raiz: .space 4
.text
sqrt:
  mv   t0, zero          // addi t0, x0, 0
  li   t1, 1             // addi t1, x0, 1
for:
  blt  a0, t1, efor
  sub  a0, a0, t1
  addi t0, t0, 1
  addi t1, t1, 2
  j    for               // jal  x0, for
efor:
  mv   a0, t0            // addi a0, t0, 0
  ret                    // jalr x0, ra, 0
main:
  la   sp, _stack        // addi sp, sp, _stack
  la   t0, num           // addi t0, t0, num
  lw   a0, 0(t0)
  call sqrt              // jal  ra, sqrt
  la   t0, raiz          // addi t0, t0, raiz
  sw   a0, 0(t0)
  j    .                 // jal  x0, .
.end
```

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

2. **(0,5 puntos)**. Justifique si es correcto o no que la función **`sqrt`** utilice únicamente registros temporales. Recodifique dicha función cambiando por preservados todos los registros temporales que sea posible.

3. **(0,5 puntos)**. Sabiendo que los registros **`t0`** y **`t1`** son respectivamente alias de los registros **`x5`** y **`x6`**, y que el código máquina de la instrucción **`addi t0,t0,1`** es **`0x00128293`** indique el código máquina de la instrucción **`addi t1,t1,2`**.

4. **(0,25 puntos)**. Suponiendo que la sección **`.text`** se ubica a partir de la dirección **`0x0`** de memoria, indique el valor del byte almacenado en la dirección **`0x11`**.

5. **(1 punto)**. Suponiendo que el programa se ejecuta en el procesador monociclo y conociendo el valor que toman en cada caso algunas de las señales de este, indique la instrucción que está realizando y el ciclo en que se encuentra (si pueden ser varios indíquelos todos).
   a) MemWr = "1"
   b) ResSrc = "00" y BRWr = "1"
   c) Jump ="1" y WA ≠ "00000"
   d) ALUctr = "001", ResSrc = "01" y BRWr = "1"

6. **(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`**. En la ruta de datos del procesador multiciclo de repertorio reducido vista en clase es posible ejecutar instrucciones **`jalr`** en 4 ciclos. Si las transferencias entre registros efectuadas en los estados S0 y S1 no pueden cambiarse y la ruta de datos tampoco, indique:
   a) Las transferencias entre registros que deben realizarse para ejecutar una instrucción **`jalr`** sobre la ruta de datos multiciclo original en cada uno de los 4 ciclos.
   b) Los valores de las señales de control ALUsrcA, ALUsrcB y ResSrc en cada uno de dichos ciclos.
   c) ¿Podría ejecutarse en menos ciclos? Justifique su respuesta.

7. **(1,5 puntos)**. Localice los riesgos que existen en la función **`sqrt`** indicando el tipo y las instrucciones involucradas en cada uno de ellos. Recodifique la función reordenando código e insertando el número mínimo de instrucciones **`nop`** para que pueda ejecutarse con corrección en un procesador segmentado sin gestión de riesgos y con escritura en banco de registros a final de ciclo.

8. **(2 puntos)**. Suponiendo que **`num`** valga 2, realice el diagrama de ejecución de la función **`sqrt`** en un procesador segmentado con gestión completa de riesgos (escritura del BR a mitad de ciclo, unidad de anticipación y unidad de conflictos con predicción de salto no tomado). 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.

9. (2 puntos). El bucle **`for`** de la función **`sqrt`** se ha implementado con evaluación al principio, pero también se puede hacer con evaluación al final como se muestra a continuación:

```
sqrt2:
  mv   t0, zero
  li   t1, 1
  blt  a0, t1, efor
for:
  sub  a0, a0, t1
  addi t0, t0, 1
  addi t1, t1, 2
  bge  a0, t1, for
efor:
  mv   a0, t0
  ret
```

Complete la siguiente tabla[1] considerando únicamente el código de la función y su ejecución con argumento 25. Suponga que todas las instrucciones de salto tienen un comportamiento análogo en las microcroarquitecturas monociclo y segmentada y que, en el caso del procesador multiciclo, la instrucción **`jalr`** tarda 4 ciclos.

	`sqrt(25)`			**`sqrt2(25)`**		
	núm. instrucciones ejecutadas	ciclos	CPI	núm. instrucciones ejecutadas	ciclos	CPI
Monociclo						
Multiciclo						
Segmentado						

a) Indique justificadamente cuál de las 2 implementaciones es más conveniente para cada microarquitectura en términos de tiempo de ejecución.
b) Para el caso de la función **`sqrt2`**, suponiendo que el tiempo de ciclo del procesador multiciclo es igual al del segmentado y que el del monociclo es cinco veces mayor, calcule el *speedup* del procesador más rápido respecto al más lento.

10. (1 punto). Suponga que un programador usa por error en la función **`sqrt`** el registro **`x1`** (cuyo alias es **`ra`**) en lugar del registro **`t1`**. Al ejecutar el programa erróneo en un procesador RISC-V se disparará una excepción, indique:

a) Tipo de excepción y justificación de por qué se produce.
b) Valor que se almacenará en el registro **`mepc`**.
c) Ciclo en que se disparará si el procesador tiene microarquitectura monociclo.
d) Ciclo en que se lanzará la primera instrucción de la rutina de tratamiento de excepción si el procesador tiene microarquitectura segmentada, suponiendo que la excepción se dispara en el ciclo i.

[1] 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). A la hora de contar los ciclos que tarda en ejecutar la función, debe contarse desde el ciclo en que lanza su primera instrucción (incluido) hasta el ciclo que se lanza la primera instrucción a la función invocante tras el retorno a esta (no incluido).