Instrucciones generales para la realización de este examen

La respuesta debe escribirse en el hueco existente a continuación de cada pregunta con **letra clara**. Cada respuesta incorrecta, ilegible o vacía no suma ni resta. En el caso de preguntas teóricas se valorará la capacidad de síntesis.

---

1 ❑ Un ordenador portatil con 4 núcleos y 8 GB RAM y una estación de trabajo con 16 núcleos y 64 GB RAM ejecutan cinco veces dos benchmarks, B1 y B2. La tabla siguiente muestra los tiempos de respuesta de cada ejecución, en segundos.

	B1					B2				
	t_1	t_2	t_3	t_4	t_5	t_1	t_2	t_3	t_4	t_5
Laptop	31.57	30.98	31.23	32.11	33.04	21.37	22.04	23.05	21.98	22.08
Workstation	17.15	18.01	17.59	17.65	18.22	15.65	14.87	14.25	14.65	15.01

a — (0.5 puntos) ¿Cuál es el rendimiento de cada computador utilizando la productividad para cada benchmark?

Laptop(B1): Workstation(B1):

Laptop(B2): Workstation(B2):

b — (0.5 puntos) ¿Cuál es la aceleración de la estación de trabajo respecto al ordenador portátil para cada benchmark?

A(B1) = A(B2) =

c — (0.5 puntos) ¿Cuál es la aceleración agregada de la estación de trabajo respecto al ordenador portátil?

2 ❑ Eva y Frank refactorizan cada uno una copia del mismo programa. Eva consigue acelerar 85 veces el 15 % del programa, mientras que Frank consigue acelerar 15 veces el 85 % del programa.

a — (0.5 puntos) ¿Cuál es la aceleración global que logra Eva en el programa? Escribe la expresión matemática que utilices para calcularla.

b — (0.5 puntos) ¿Cuál es la aceleración global que logra Frank en el programa? Escribe la expresión matemática que utilices para calcularla.

3 ❑ Se ejecuta el siguiente programa MIPS64 sobre una microarquitectura segmentada básica cuya única mejora es una unidad de multiplicación/división de enteros de 5 ciclos no segmentada. Las excepciones precisas no están soportadas.

```
1     ori r5, r0, 3 ; r5 = 3
2     ddiv r2, r4, r5
3  dest:
4     daddi r5, r5, -2 ; r5 = r5 - 2
5     beq r5, r0, dest ; branch on r5 equal to r0
6     or r3, r0, r9
```

a — (1 punto) Enumera las dependencias de datos que existen en el programa, especificando tipo junto con las instrucciones y registros involucrados. **Ejemplo de respuesta**: RAW => `daddi, ori : r7` // `dsub, xori : r3`

RAW =>

WAW =>

WAR =>

b — (1 punto) Indica las duraciones de las detenciones expresadas en ciclos de reloj. Para los tipos RAW, WAW y estructural debe indicarse además la instrucción que se detiene, mientras que para el tipo control la instrucción que la provoca. Si un tipo de detención no aparece, indícalo con **N/A**. Si hay varias detenciones del mismo tipo deben indicarse todas.
Ejemplo de respuesta, RAW: `xor` 1 ciclo // `andi` 2 ciclos

RAW:

WAW:

Estructural:

Control:

c — (1 punto) ¿Cuántos ciclos de reloj son necesarios para ejecutar el código anterior? ¿Cuál es el CPI ignorando el transitorio inicial? **Indica la expresión matemática** utilizada para obtener el CPI.

Ciclos:	CPI:

d — (0.5 puntos) Si todas las rutas de reenvío estuviesen implementadas en esta microarquitectura, ¿qué rutas se activarían al ejecutar el programa anterior? **Ejemplo de respuesta**: Salida `EX daddi` –> Entrada `EX dmul`.

e — (1 punto) Si se implementase renombrado de registros en esta microarquitectura, ¿qué registros arquitectónicos cambiarían de registro físico al final de la ejecución del programa y a qué registro físico estarían asignados? Para el renombrado se dispone de una cola FIFO que originalmente contiene los registros `rr32` a `rr63` y a los que se van añadiendo los registros disponibles
Ejemplo de respuesta para esta pregunta: Registro `r7` asignado a `rr43`.

4 ❑ Una microarquitectura MIPS64 implementa **evaluación clásica de saltos** (en la etapa `MEM`), predición de saltos siempre no tomado y **sin rutas de reenvío**. Sobre esta microarquitectura se ejecuta el siguiente código.

```
1         daddi   r1, r0, 32
2  start:
3         beqz  r1, end  ; Branch on Equal Zero
4         daddi r1, r1, -4     ; r1 decrement by -4
5         dsub  r3, r7, r6
6         j     start      ; Jump to start
7  end:
8         sd    r3, 100(r0)
9         xor   r3, r3, r3
```

a — (1.5 puntos) Rellena **todas las filas** del siguiente cronograma con la evolución de las primeras instrucciones del programa sobre el *pipeline*.

Instrucción \ Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`daddi r1, r0, 32`	IF													
`beqz r1, end`														
-														
-														

b — (0.5 puntos) Teniendo en cuenta las dos instrucciones de salto del programa. Para el programa completo, ¿Cuántas veces en total acierta la predicción el predictor siempre no tomado? ¿Y cuántas veces en total falla la predicción?

Núm. aciertos:	Núm. fallos:

c — (0,5 puntos) Si el predictor de saltos siempre no tomado se sustituye por un predictor dinámico de 2 bits con el valor `01` por defecto del historial (*weak not taken*), ¿cuántos ciclos de reloj se detendría el *pipeline* en la ejecución del **programa completo** debido a la instrucción `beqz r1, end` y a la instrucción `j start`?

`beqz r1, end`:	`j start`:

d — (0,5 puntos) Si el programa se ha cargado a partir de la dirección de memoria `B63Ch` ¿Cuáles serán los valores de las entradas en la tabla BHT+BTB correspondientes a las instrucciones de salto al finalizar la ejecución del programa?

Dirección	Destino	Historial