General instructions

Answers must be written in the boxes reserved after each question. Each wrong, illegible or blank answer does not increase or decrease the final mark. In the case of theoretical questions, the capacity for synthesis will be assessed.

---

1 ❑ A laptop with a 3-core CPU and 6 GB RAM and a workstation with 14 cores and 32 GB RAM run five times two benchmarks: B1 and B2. The table shows the response time of each execution, in seconds.

	B1					B2				
	t_1	t_2	t_3	t_4	t_5	t_1	t_2	t_3	t_4	t_5
Laptop	20.87	21.15	20.98	21.45	22.06	23.1	22.7	21.4	21.6	23.4
Workstation	4.15	4.19	4.65	4.37	3.99	1.11	1.12	1.14	1.17	1.51

a — (0.5 points) What is the performance of each computer, using the throughput, for each benchmark?

Laptop(B1): Workstation(B1):

Laptop(B2): Workstation(B2):

b — (0.5 points) What is the speedup of the workstation compared to the laptop for each banchmark?

A(B1) = A(B2) =

c — (0.5 points) What is the aggregated speedup of the workstation compared to the laptop?

2 ❑ Alice and Bob refactor a copy of the same program. Alice speeds up 95 times 5 % of the program, whereas Bob speeds up 5 times 95 % of the program.

a — (0.5 points) What is the overall speedup achieved in the program by Alice? Write down the mathematical expression used to compute the result.

b — (0.5 points) What is the overall speedup achieved in the program by Bob? Write down the mathematical expression used to compute the result.

3 ❑ The next MIPS64 program is run on a basic pipelined microarchitecture with the following improvements: a non-pipelined 2-cycle integer mult/div unit and early branch evaluation in `ID`. Precise exceptions are not supported.

```
1     bne r4, r4, next ; branch on registers not equal
2     andi r3, r4, 90
3  next:
4     dmul r2, r6, r3
5     ld r6, 20(r8)
6     ori r2, r6, -1
```

a — (1 point) Write the data dependencies of the above program, specifying the type, instructions and registers involved. **Example of an answer** for this question: RAW => `daddi, ori : r7 // dsub, xori : r3`

RAW =>

WAW =>

WAR =>

b — (1 point) Write the duration, expressed in clock cycles, of the following types of stalls if they appear in the execution of the previous snippet. For RAW, WAW and structural stalls you should write the stalled instruction also while for control stalls the instruction responsible. If any type does not appear, write **N/A**. If there are several stalls of the same type, all must be indicated. **Ezample of an answer** for this question, RAW: `xor` 1 cycle // `andi` 2 cycles

RAW:

WAW:

Structural:

Control:

c — (1 point) How many clock cycles does it take to run the above code if the CPU? What is the CPI factor ignoring the initial transient period? Write down the **mathematical expression** used to obtain this factor.

Cycles:	CPI:

d — (0.5 points) If all possible forwarding paths are implemented in this microarchitecture, which paths will be activated while running the program? **Example of an answer** for this question: Output `EX daddi` –> Input `EX dmul`.

e — (1 point) If register renaming is implemented in this microarchitecture, what architectural registers will change of assigned physical register at the program? In order to select a physical register for renaming, a FIFO queue is available which originally contains registers `rr32` to `rr63`. **Example of an answer** for this question: Register `r7` is assigned to `rr43`.

4 ❑ A MIPS64 microarchitecture with **standard branch evaluation** (in the `MEM` stage), always-not-taken branch prediction, and no forwarding paths runs the following code snippet.

```
1        ori   r12, r0, 44
2  loop:
3        beq  r12, r0, endloop  ; Branch on Equal
4        daddi r12, r12, -4     ; r12 decrement by -4
5        dsub  r10, r11, r16
6        j     loop      ; Jump to loop
7  endloop:
8        sd    r10, 100(r0)
9        xor   r10, r10, r10
```

a — (1.5 points) Fill **all the rows** of the following chronogram with the evolution in the pipeline of the first instructions.

Instruction \ Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14
`ori r12, r0, 44`	IF													
`beq r12, r0, endloop`														
-														
-														

b — (0.5 points) Taking the two control instructions of the program, in the whole program how many times does the always-not-taken predictor hit? And how many times does the predictor miss? (The predictor is only applied to conditional branch instructions).

NÃºm. aciertos:	NÃºm. fallos:

c — (0,5 pooints) If the always-not-taken branch predictor is substituted by a 2-bit dynamic predictor, with default value `01` (*weak not taken*) for historial, how many clock cycles would the pipeline be stalled in the execution of the **whole program** due to the `beq r12, r0, endloop` instruction? And due to the `j loop` instruction?

`beq r12, r0, endloop`:	`j loop`:

d — (0,5 points) If the code is loaded in memory from address ÂB5CChÂ, what are the values of the BHT+BTB table associated with the branch and jump instructions at the end of the execution?

Address	Target	Historial