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 2 cores and 6 GB RAM and a workstation with 4 cores and 32 GB RAM run five times the same monothreaded benchmark. The table below shows the response time of each running, in seconds.

	t_1	t_2	t_3	t_4	t_5
Laptop	20.87	21.15	20.98	21.45	22.06
Workstation	4.15	4.19	4.65	4.37	3.99

a — (0.25 points) What is the performance of each computer using throughput?

Laptop: Workstation:

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

c — (0.75 points) Now, a multithreaded version of the benchmark is run. It uses 8 threads. The response time of the benchmark in each computer is the one shown in the above table divided by the number of threads that can run concurrently. What is the speedup of the workstation compared to the laptop?

2 ❑ Given a 4-stage pipelined CPU with a 0.5 ns clock period,

a — (0.5 points) What is its maximum throughput, expressed in MIPS?

b — (0.5 points) If the CPU were, in addition, superscalar with an issue width of 3, what would be its maximum throughput, expressed in MIPS?

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

```
1     ori r5, r0, 5 ; r5 = 5
2     xor r1, r4, r5
3     bnez r5, label ; branch on r5 not equal to zero
4     slt r4, r8, r7
5  label:
6     dmul r3, r8, r6
7     andi r3, r0, 7
```

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   r1, r0, 30
2  start:
3        beqz  r1, end  ; Branch on Equal Zero
4        dsub  r2, r3, r4
5        daddi r1, r1, -3     ; r1 decrement by -3
6        j     start      ; Jump to start
7  end:
8        sd    r2, 100(r0)
9        xor   r2, r2, r2
```

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 r1, r0, 30`	IF													
`beqz r1, end`														
-														

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).

# of hits:	# of misses:

c — (0,5 points) 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 `beqz r1, end` instruction? And due to the `j start` instruction?

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

d — (0,5 points) If the code is loaded in memory from address ÂA730hÂ, 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