1.

a. We know that `ORI` is an I-type instruction. I-types require RegWrite and AluSrc
Accessing PC: 20 ps
Accessing Instruction: 310 ps
Reading Register File: 170 ps
Immediate Generation: 40 ps
Control: 75 ps
Register Write: 12 ps
ALU: 240 ps

Immediate Generation and Control both run PARALLEL to read the register. So we choose the maximum of the three which is read register and we add everything else (that runs in series)!

Accessing PC + Accessing Instruction + Reading Register File + Register Write + ALU = 752 ps

b. `LW` instruction is also I-type instruction. This requires RegWrite and AluSrc
Accessing PC: 20 ps
Accessing Instruction: 310 ps
Reading Register File: 170 ps
ALU: 240 ps
Read Data From Memory: 310 ps
Mux: 20 ps
Register Write: 12 ps

The updating PC path is far shorter than the one I just listed so we don't really need to consider the other one. This path's speed is equal to 1082 ps however.

c. `BEQ` is a B-type instruction. This requires branching!
Through Registers:
Accessing PC: 20 ps
Accessing Instructions: 310 ps
Reading Register File: 170 ps
Mux: 20 ps
ALU: 240 ps
AND Gate: 3 ps
Mux: 20 ps
PC Write: 10 ps

Through Adder:
Accessing PC: 20 ps
Adder: 140 ps
Mux: 20 ps

The longer path is obviously through registers, the sum gives us 793 ps.

d. This is the maximum of all instructions. Running max(`ORI`, `LW`, `BEQ`) = 1082 ps

2.

a. We can split each stage into their own 5 separate little categories:
Instruction Fetch (IF):
Accessing PC: 20 ps
Accessing Instructions: 310 ps
Total: 310 ps

Instruction Decode (ID):
Reading Register File: 170 ps
Immediate Generation: 40 ps
Control Unit: 75 ps
But these 3 run in parallel, so we choose the longest which is 170 ps

Execution/ALU (EX):
ALU: 240 ps
Adder: 140 ps
ALU and Adder work in parallel so we choose 240 ps

Memory Access (MEM):
Memory Read/Write: 310 ps

Write Back (WB):
Register Write: 12 ps

Looking at the longest time our system will be bottlenecked by the longest stages, so it would be bottlenecked by IF and MEM which are both 310 ps.

b. `ORI` goes through all stages only once. The instructions are fetched from memory (IF), they are decoded (ID)–`rs1` is read, immediate generated, etc.–, ALU is done (EX), the operation still goes through MEM despite the fact we don't actually

need it, and the result of the operation is written back to the destination in WB. Since we have 5 cycles and 310 ps clock time, our total latency is 1550 ps.

c. Since we don't actually use MEM we only have four cycles with the time of 310 ps in each. So our total time is 1240 ps.

d. For `LW`, the total latency would remain the same since we still have to pass through all five stages and they each have a latency of 310 ps (because we are still gatekept by the maximum latency) and the total results in 5 * 310 ps = 1550 ps