15 POINTS

3. On an x86 single core **Linux** platform, when the **Initial Thread** (IT) of a newly created process (a process just created by a **fork()** call) begins executing in user space for the **very first time**, it always encounters an **immediate TLB fault**, even though it may then get **an L1 hit** after updating its TLB via a table walk.

A. **Explain why** the thread encounters an **immediate TLB miss.**

Since the thread is in a different process (HW context switch) a TLB shootdown invalidated the previous entries

B. **Explain** how it's possible for the thread to get **an L1 hit** after filling the missed TLB entry (via the table walk mechanism).

Because the forked process is running the same code as its parent, and since the parent was just running its memory would still be in cache

C. Is there likely to be a **page fault or context switch** involved in the sequence described above (i.e. the TLB miss, followed by table walk, followed by L1 hit) ?? **Explain.**

No, because the memory left by the parent is still valid for the child at this point, neither page faults nor context switches happen during the process described