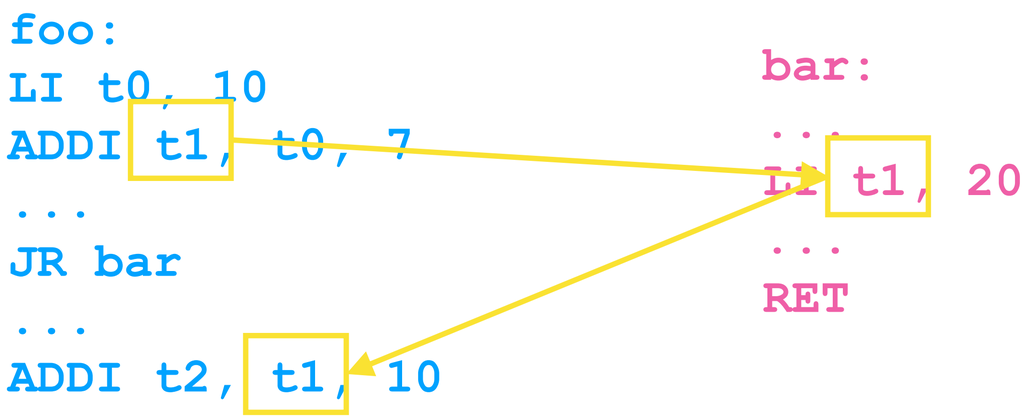
What happens if foo and bar reuse registers?

two functions

- Ignoring machinery for stack/return address/frame management



- What's wrong with this code?

saving registers

- To avoid overwriting registers, it is important to save all registers that the caller is using **and** the callee will overwrite
  - Careful about "using": a caller needs a register if the value it has *before* the callee is invoked is used *after* the callee is returned
  - More precisely, the register is **live** across the function call (we will define this more carefully in a later lecture)
- Save registers onto the stack when making a call
- Restore registers from the stack when returning from a call

Problem

- Do not know the caller/callee relationship!
  - Caller may not know all possible functions callee invokes → cannot tell exactly which registers will be overwritten
  - Callee may not know who calls it → cannot tell exactly which registers are in use

callee saves vs. caller saves

- Can be conservative and save extra registers
  - Callee can save all registers it overwrites even if the caller doesn't use them: **callee saves**
  - Caller can save all registers it uses, even if callee does not overwrite them: **caller saves**
- Who saves the registers determines which activation record holds the registers
  - Callee saves: put saved registers on stack before allocating space for locals, restore them on return
  - Caller saves: put saved registers on stack before allocating space for arguments and return values, restore them on return
    - Question: why not put saved registers on stack *after* arguments and return values?

ABI

- Determining what register saving convention to use is part of a system's **application binary interface**
  - All software written for an architecture/OS should use the same convention
  - What happens if not?
- Can use some combination of caller saves and callee saves
  - Risc-V dictates that some registers are the caller's responsibility to save, and some registers are the callee's responsibility to save

- In project, we will always use callee saves: save all registers written by the callee
  - One exception: RA gets overwritten by JR, so caller must save it