Register Conventions (1/4)

- CalleR: the calling function (where you call a function)
- CalleE: the function being called
- When callee returns from executing, the caller needs to know which registers may have changed and which are guaranteed to be unchanged.
- Register Conventions: A set of generally accepted rules as to which registers will be unchanged after a procedure call (`jal`) and which may be changed.

Register Conventions (2/4) – saved

- **`$0`**: No Change. Always `0`.
- **`$s0`-`$s7`**: Restore if you change. Very important, that's why they're called saved registers. If the callee changes these in any way, it must restore the original values before returning.
- **`$sp`**: Restore if you change. The stack pointer must point to the same place before and after the `jal` call, or else the caller won't be able to restore values from the stack.
- HINT -- All saved registers start with S!

It's callee's job to restore!

Register Conventions (3/4) – volatile

- **$ra**: Can Change.
  - The jal call itself will change this register. Caller needs to save on stack before next call (nested call).
- **$v0-$v1**: Can Change.
  - These will contain the new returned values.
- **$a0-$a3**: Can change.
  - These are volatile argument registers. Caller needs to save if they are needed after the call.
- **$t0-$t9**: Can change.
  - That's why they're called temporary: any procedure may change them at any time. Caller needs to save if they'll need them afterwards.

It's caller's job to backup!

Register Conventions (4/4)

- What do these conventions mean?
  - If function **R** calls function **E**, then function **R** must save any **V (volatile) registers** that it may be using onto the stack **before making** a **`jal`** call.
  - Function **E** must save any **S (saved) registers** it intends to use before garbling up their values. It must restore any modified **S** registers **before returning** back to **R**
- Remember: caller/callee need to save only volatile/saved registers they are using, not all registers.