LOAD, STORE, LVALUES.

If you have a global variable that happens to occupy memory location number 26, and a local variable that happens to live 2 locations below the frame pointer, then to load the global into register 1 and the local into register 2, we would expect something like this:

\[
\begin{align*}
\text{LOAD} & \quad R1, * 26 \\
\text{LOAD} & \quad R2, * FP - 2 \\
\end{align*}
\]

OK? A mere “LOAD R1, 26” would put the number 26 into register 1, not the value read from memory locations 26.

Because of that, we would naturally expect that storing register 1 in the global variable and register 2 in the local would be exactly identical-ish.

\[
\begin{align*}
\text{STORE} & \quad R1, * 26 \\
\text{STORE} & \quad R2, * FP - 2 \\
\end{align*}
\]

In order for this regularity to work, the STORE instruction must work in an irregular way. If your fetch-execute cycle looks something like this:

```c
int operand_value = operand;
if (reg2 != 0) operand_value += register_array[reg2];
if (indirect != 0) operand_value = read_mem(operand_value);
switch (opcode)
{
.....
    case OP_LOAD: register_array[reg1] = operand_value;
        break;
    case OP_ADD: register_array[reg1] += operand_value;
        break;
    case OP_MUL: register_array[reg1] *= operand_value;
        break;
    case OP_STORE: ..... what?? ..... = register_array[reg1];
        break;
}
```

Most instructions work in exactly the same way, but STORE needs the operand_address, not the operand_value. In fact these two little bits:

\[
\begin{align*}
\text{STORE} & \quad R1, 26 \\
\text{STORE} & \quad R2, R3 \\
\end{align*}
\]

Just won’t have any meaning.

Either that, LOAD and STORE will have to be used in different ways. One way or the other, something has to be irregular and slightly confusing.

**BOTH WAYS ARE OK.** This is pointing out a frequent source of confusion. If you are happy with “LOAD R1, * 26” having “STORE R1, 26” as its functional counterpart instead of the visually matching “STORE R1, * 26”, that is perfectly alright, you do not need to change anything. Just take care not to get confused at any point.