```
// register/name based
y = (x + 5) * 3

// previous IR
t = x + 5
y = t * 3

// or explicitly

t1 = // load the local x
t2 = // constant 5
t3 = t1 + t2
t4 = // constant 3
t5 = t3 * t4
// store t5 into local y
```

```
// stack based
load(x) // push value of local x on to the stack
constant(5) // push constant 5 on to the stack
add // pop the top two values, adds, and pushes result on to the stack
constant(3) // push constant 3
mul // pop the top two values, multiplies, pushes result
store(y) // pop top value, store it in the local y
```

```
// interpreter loop
while bytecode != nullptr {
        switch (bytecode) {
        case bytecode is "load x":
                t = local_variables[x];
                push(t); // on to stack
                break;
        case bytecode is "add":
                b = pop();
                a = pop();
                t = a + b;
                push(t);
                break;
        case ...:
        }
}
```