Autocode - automatic generation of assembly code

We should be able to type a file like this:

```plaintext
function fff x y
    local a b c
begin
    set a = + * 3 * x * x + * 5 * x 1
    set b = + * 7 * y * y * y + * 4 * y * y + * 9 * y 2
    set c = * a b
    return c
end

function main
    local cat bat hat
begin
    set cat = + 1 2
    set bat = + 3 4
    call fff 2 cat bat -> hat
    print hat
end
```

and have our autocode translator reliably produce correct assembly code for it.

A line only says one thing, and says it in a very simple way.

A line like `function fff x y`
- adds an entry `fff=0` to the symbol table
- adds an entry `x=+2` to the symbol table
- adds an entry `y=+3` to the symbol table
- starts the count of local variables to 0
- produces the output `fff:
- produces the output `push fp`
- produces the output `load fp, sp`

A line like `local a b c`
- adds entries `a=-1`, `b=-2`, `c=-3` to the symbol table
- adds three to the count of local variables

Invent an easy syntax for introducing large things like arrays, maybe `local array b 9` Also remember that you’ll want globals too.

The line `begin`
- produces the output `sub sp, localvariablecount`
The line `end`
produces the output `load sp, fp`
produces the output `pop fp`
produces the output `ret`
removes all locals and parameters from the symbol table

A line like `return` or `return value`
if there is a value, uses the polish converter to put it in register 0
produces the output `load sp, fp`
produces the output `pop fp`
produces the output `ret`

A line beginning with `set`
uses the polish converter to get the value in R1
produces the output `store r1, [thedesination]`

and so on and so on and so on.

Design your own autocode language and create a translator for it. This has to be a small-step by small-step process. Get something very basic working properly, and only add small amounts to things that already work.