CSC301 Assignment #3

Problem 4.4

Exercise 4 Suppose the target assembly language for a compiler has these five instructions for integers:

```
load address, reg
add reg, reg, reg
sub reg, reg, reg
mul reg, reg, reg
store reg, address
```

In these instructions, an *address* is the name of a static variable (whose actual address will be filled in by the loader). A *reg* is the name of an integer register, a special extra-fast memory location inside the processor. The target assembly language has three integer registers: `r1`, `r2`, and `r3`. The `load` instruction loads the integer from the given memory address into the given register. The `add` instruction adds the second register to the first register and places the result in the **third** register. The `sub` instruction subtracts the second register from the first register and places the result in the third register. The `mul` instruction multiplies the first register by the second register and places the result in the third register. The `store` instruction stores the integer from the given register at the given memory address. So, for example, the compiler might translate the assignment `result := offset+(width*n)` into this:

```
load width,r1
load n, r2
mul r1, r2, r1
load offset, r2
add r1, r2, r1
store r1, result
```

Using this assembly language, give translations of the following assignment statements. Use as few instructions as possible.

a. `net := gross - costs`
b. `volume := (length * width) * height`
c. `cube := (x * x) * x`
d. `final := ((a - abase) * (b - bbase)) * (c - cbase)`

(Many modern microprocessors implement a *load/store architecture* like this, though usually with more registers.)