Assembly Language Homework

Problem 1

In class we developed a small assembly language for a register-based CPU. As a refresher, here is our program that computes: c = a + b

```
a: 58
b: 13
c: 0

.code
load a, r1
load b, r2
add r1, r2, r3
store r3, c
```

In addition to `add` assume that our CPU has instructions `sub`, `mul`, and `div` for subtraction, multiplication, and division. Also assume that our CPU has a total of 4 registers r0--r4. The arithmetic operations can be done on any 3 registers, and the "output" always goes in the third register listed. So these are legal instructions:

```
add r1, r2, r1
add r1, r1, r1
```

For each part below, show a complete sequence of instructions to perform the given computation. Try to use as few instructions as possible. Note: you do not have to show the data section, just write the .code section.

a. `profit = sales  - expenses`

b. `vol = (len * width) * height`

c. `discrim = b * b - 4 * a * c`

d. `dist2 = ((x2 - x1) * (y2 - y1)) * (z2 - z1))`

Problem 2

Some CPUs have a stack-based architecture instead of registers. Consider an alternative assembly language with these operations:

```
push address   -- push the value at the given location onto the stack
add            -- pop two values from stack, add them, and push result
sub            -- like add, but first item popped is subtracted from second
mul            -- like add, but multiplies
div            -- like sub, but divides
pop address    -- pops top value from stack and stores at the given location
```

Using this assembly language, the code for our simple class example looks like this:

```
a: 58
b: 13
c: 0

.code
push a
push b
add
pop c
```

Repeat parts a--d of Problem 1 using this alternative assembly language.