CSC301 Assignment #3
Baheem Ferrell

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: rl, 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,rl
 load n, r2
 mul rl, r2, rl
 load offset, r2
 add r1, r2, r1
 store rl, result
```

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

a. net := gross – costs

since - sub instruction would be used
assembly-
load r1,gross //R1 = GROSS
load r2, costs //R2 = COSTS
sub r1,r2,r1 //R1 = GROSS-COSTS
store r1, net //NET = R1

b. volume := (length * width) * height

mul operation will be used
assembly-

load r1,length //R1= LENGTH
load r2, width //R2 = WIDTH
mul r1,r2,r1 //R1 = LENGTH*WIDTH
load height , r2 //R2 = HEIGHT
mul r1,r2,r1 //R1 = (length*width)*height
store r1,volume //VOLUME = R1

c. cube:=(x*x)*x

same as above just multiple load are not required
assembly-
load x,r1 //R1 = X
mul r1,r1,r1 //R1 = X*X
mul r1,r1,r1 //R1 = (X*X)*X
store r1,cube //CUBE = R1

d. final := ((a – abase) * (b – bbase)) * (c – cbase)

assembly
load a ,r1 //R1 = A
load abase,r2, //R2 = ABASE
sub r1,r2,r1 //R1 = A-ABASE
load b,r2 //R2 = B
load bbase,r3 //R3 = BBASE
sub r2,r3,r2 //R2 = B - BBASE
mul r1,r2,r1 //R1 = ((A-ABASE)*(B-BASE))
load c,r2 //R2 = C
load cbase , r3 //R3 = CBASE
sub r2,r3,r2 //R2 = C- CBASE
mul r1,r2,r1 // ((A-ABASE)*(B-BASE))*(C- CBASE)
store r1,final //FINAL = R1

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