Lab 4: Chemical Formula

Professor Patt hates chemistry. When he gets to a chemical formula, he often cannot figure out its structure and count how many atoms are in it. As a computer man, he would like to have a program to help him. The program can count the number of each kind of atoms in the chemical formula, turning it into a molecular formula.

Your job is to write a program to convert the input chemical formula into a molecular formula.

> This lab is associated with the following chapters and sections. If you have no idea how to do this lab, please review the corresponding parts of the book.
>
> - 8.2 The Stack
> - 8.5 Character Strings
> - 10 A Calculator

Implementation Details

- You are required to write in **LC-3 assembly language**.
- Your program should start at x3000, which means the first instruction of your program is located in position x3000.

The input chemical formula

The input *chemical formula* contains:

- `Xn`, where `X` is the symbol of the chemical element, `n` is the number of this element. To make it easy, the length of `X` is limited to 1 character, that is, the symbol is from `A` to `Z`.
- `(...)n`, where `...` surrounded by brackets is a *polyatomic ion* (a group of atoms), `n` is the number of this polyatomic ion. There may be nesting brackets in a pair of brackets.
- In both cases, if `n` is 1, it can be omitted. `n` ≠ 0.

The output molecular formula

The output *molecular formula* should be like:

- Print each kind of chemical element and its amount as `Xn` in alphabetic order of the element symbol.
- If the amount is 1, `n` should be omitted.

Hint

- A **stack** should be used to parse the nesting brackets.
- Since the input chemical formula only contains elements `A` to `Z`, an **array** of length 26 can be used to store the amount of each element.

How to Run?

We provide you 4 with useful trap routines to perform multiplication, division, and conversion between ASCII string and number. Note that these functions only support positive numbers.

	vector	function
MUL	x80	Multiply R0 and R1. Store the result in R0.
DIV	x81	Divide R0 by R1, Store the quotient in R0, and the remainder in R1.
PARSE_INT	x82	Parse an ASCII string starting from R1 to a number. Store the number in R0, and move R1 to the address of the next non-digit character.
PRINT_INT	x83	Print R0 as a number to the console.

The simulator does not know what `MUL` is, so you have to write `TRAP x80` when you want to use the multiplication. In order to make these trap routines available, you should:

1. Reinitialize or randomize the machine.
2. Load the provided trap routines `lab4traps.obj` in to the simulator.
3. Load your program into the simulator. And run the machine.

- Do not paste the contents from `lab4traps.asm` to your code.
- If you do not want to use these functions, just ignore them and write new subroutines in your code.
- Do not overwrite these trap routines.

Samples

```
CH3CH2CH2CH3
C4H10

--- Halting the LC-3 ---
```

```
HOOCCH2C(OH)(COOH)CH2COOH
C6H8O7

--- Halting the LC-3 ---
```

```
CH3(CH2)16CH3
C18H38

--- Halting the LC-3 ---
```

In each sample, the first line is the input and the second line is the output.

Limitations

- The length of the input chemical formula l: $0 \leq l < 100$.
- The number of each kind of atoms n in the output molecular formular: $0 \leq n < 2^{15}$.
- The number of types of chemical elements x: $0 \leq x < 26$.
- Acceptable program length: less than 300 lines.

Grading

Lab 4 takes 0% of the total score.