545 SW Architecture — Calc Net

Project #2 – Calc Net
Introduction

This project is to write an Html+Javascript program to run a factory floor of machines which help compute an assignment statement. Each machine can handle a part of the work. The statement right-hand-side (RHS) is an arithmetic expression using addition, multiplication, and exponentiation, integer constants in [0..9], and single-letter variables.

Sample Input String

"Z=5*X^2+7*X*Y+3*Y^2+1."
- o- The end of input is marked with a period mark
- o- The only operations are those shown in this sample
- o- The variables are the 26 uppercase letters
- o- No parentheses, or other operators are allowed
- o- There are no spaces in the statement string.

Machines

The machines are of 6 kinds: I, A, E, T, P, and D. They can send blocking messages to each other. When a machine is working on a message, it is not available for other work and replies with a NAK: not acknowledged. A machine accepting a work message first acknowledges the message with an ACK reply, then works to produce the result, then sends a result message to the original sender. A machine is busy until it either receives a NAK or it receives the result message. While it is busy, it returns other request messages with a NAK.

A = Assignment machine. It can take an assignment statement (as above), split off the RHS and send it to an E machine, get the E result value, and send a Store message to a D machine. On getting the Stored result message, the A machine returns the value to the I machine and becomes ready for another statement.

E = Expression machine. It can take an arithmetic expression (a polynomial as above) and split off each term and send each term to a T machine and get the result integer back and sum all the results from all the terms and return that sum to its sender. It has an array large enough to store the value of each term in the polynomial it gets (to simplify its life, and help make visible what is going on).

T = Term machine. It can take a term (of multiplied factors) and split off each factor. If the factor is a constant, it can be used directly. If the factor is a variable, it can send it to a D machine and get the result integer value back. If the factor is an exponentiation, it can send this to a P machine and get the result integer value back. It can then compute the product of all these values and return the result to its sender. It also has an array for temporary storage of the various factors it splits off.

P = Power machine. It can take a power expression consisting of a variable and a constant integer exponent. It returns the result to its sender. It sends a Load message with the variable to a D machine, then uses the result value to compute the exponential value and return it to the sender. It has a 2-element temporary array.

D = Data machine. It can take two messages. The first is a Load message with the name (a letter) of a variable. It returns the variable's current value to the sender. The second is a Store message with both the name of variable and a constant value. It stores the value in the variable's box (a slot in an internal array), and returns a STORED message. It has an internal array for 26 variables.

I = Input machine. This machine has the next 5 input statements to be computed. It sends each statement in a message to the A machine. It puts the result value next to the current statement and then makes the next statement current.

For this project, we have two copies of the T machine, but only one copy of the other kinds.

Display

Each machine should have a panel (AKA an outlined box area) on the display. When a machine gets a message, that message is displayed in its panel. If the machine has internal storage, it show should the current state of that storage.

Each machine should have a visible connection (e.g., a line, an arc) to each other machine that it can reasonably send a message to (check the above machine descriptions).

The I machine panel is a bit special in that it holds a fixed set (max = 5) of input statements for a run, one line per statement. Each statement is run by sending it to an A machine. The result value is displayed next to the statement. The current statement is highlighted. Some of a run's statements can be simple variable setup assignments (e.g., X=12. or Y=3.). Either allow the up-to-5 statements to be read from a fixed file, or allow them to be entered via the HTML web page, whichever seems easier.

Architecture Analysis

You should prepare a 1-page (at most) paper describing your architecture using the concepts described in class.

Team

The team size is the same as before, but you can change team members from the previous project if you wish.

Project Reports

As before, but simplified a bit. Please check out the updated sample Report pdf.

Readme File

As before.

Academic Rules

Correctly and properly attribute all third party material and references, if any, lest points be taken off.

Submission

As before.

Grading

As before.