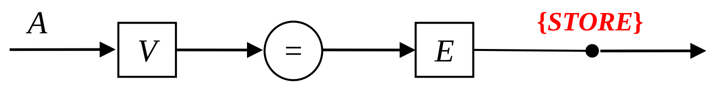
PROJECT

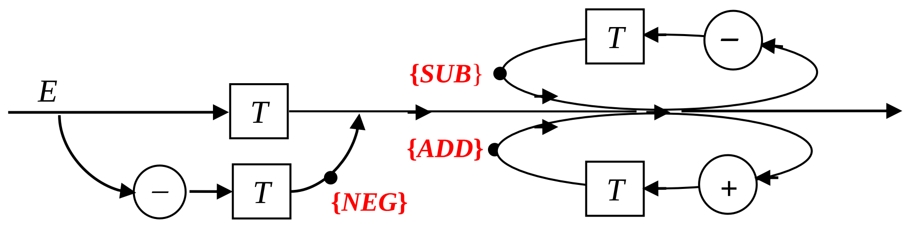
(Details will be discussed in Lecture)

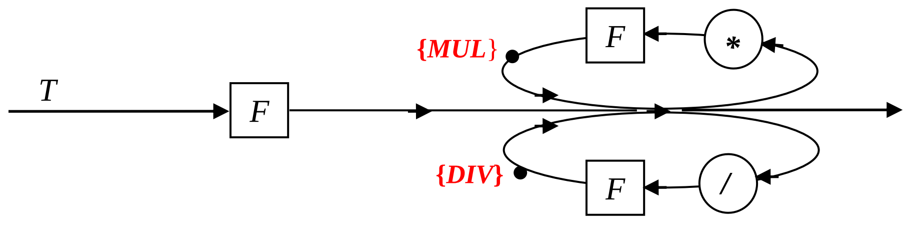
Write a recursive-decent compiler for the for translation of assignment statements into an assembly language for a stack machine, having zero-address instructions and using an evaluation stack for carrying the computations.
The instruction set of the target machine is defined as follows :

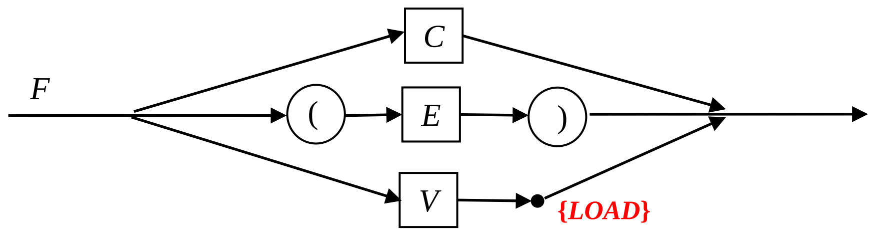
Instruction	Meaning
LIT	Pushes next address or constant onto the stack.
LOAD	(assumes an address on top of stack), replaces address on top of stack by the referenced value.
STORE	(assumes top element of stack is constant and the next to top is address), stores the constant at the cell referenced by the given address, removes the two elements from stack.
NEG	(assumes a constant on top of stack), changes the sign of the top element on stack.
ADD, SUB MUL, DIV	Operates on two top elements on stack by the specified operation, removing them and then placing the result back on stack.

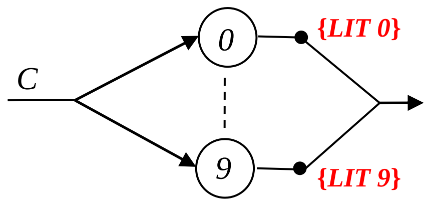
The source language text is defined by the following syntax diagrams. In the Syntax diagrams, the place of inserting semantic actions is marked by thick point, and the generated code is written in red.

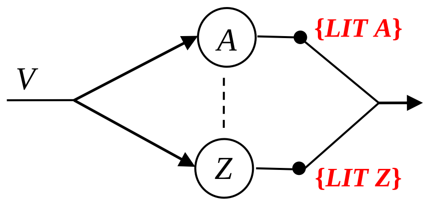












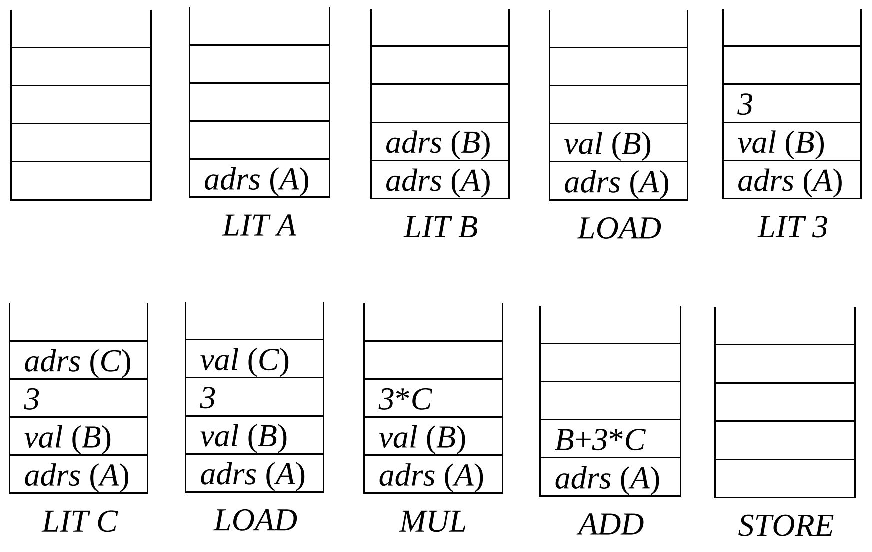
For example, the source string "

A = *B* + 3 * *C*

is to be translated to :

LIT A LIT B LOAD LIT 3 LIT C LOAD MUL ADD STORE

Tracing of the computation :


adrs (A)
LIT A
adrs (B)
adrs (A)
LIT B
val (B)
adrs (A)
LOAD
3
val (B)
adrs (A)
LIT 3
adrs (C)
3
val (B)
adrs (A)
LIT C
val (C)
3
val (B)
adrs (A)
LOAD
3*C
val (B)
adrs (A)
MUL
B+3*C
adrs (A)
ADD
STORE