2011 Paper 3 Question 5

Compiler Construction

Consider a simple grammar for arithmetic expressions:

$$E ::= n \mid x \mid -E \mid E \ + \ E \mid (\ E\ )$$

with n ranging over integer constants and x ranging over variables. We want to compile expressions to code for a simple stack-based machine with the following instruction set.

instruction	meaning
`pushvar` k	push the value of the k-th variable on top of stack
`push` n	push n on top of stack
`add`	replace the top two stack items with their sum
`neg`	replace the top stack item with its negation

For this problem, we will not worry about how variables are bound to values nor how abstract syntax trees are produced.

(a) How will your compiler generate code from expressions of the form $-E$?
[4 marks]

(b) How will your compiler generate code from expressions of the form $E_1 \ + \ E_2$?
[4 marks]

(c) What code will your compiler generate for the expression `-(-x + (17 + y))`?
[4 marks]

(d) Suppose we now want to extend the language of integer expressions with multiplication

$$E ::= \cdots \mid E \ * \ E$$

but we cannot extend the machine with an instruction for multiplication.

Can you implement this extended language directly with the machine instruction set presented above? If not, suggest a *minimal* extension to the instruction set that allows for the implementation of multiplication using the addition from the instruction set. Explain the semantics of your extensions and how you would use them to implement multiplication. [8 marks]