Compilers

* Given a context-free grammar, G, and the grammar-independent functions for a recursive-descent parser, complete the recursive-descent parser by adding the grammar-dependent functions.
  + Formal Definition
    - G = (V, Ʃ, P, S)
    - V = variables a finite set
    - Ʃ = alphabet or terminals a finite set
    - P = productions a finite set
    - S = start variable S €V
    - Productions’ form where A€V, ɑ €(V Union Ʃ)\*”
    - A -> ɑ
  + Example
  + G =
    - S -> NP VP
    - NP -> N | Adj NP
    - N -> car | dog
    - Adj -> big | green
    - VP -> V | V NP
    - V -> is | eats
    - Example Derivation:
    - S =>G NP VP =>G Adj NP VP =>G Adj N VP =>G Adj N V =>G Adj N **eats** =>G **big** N **eats** =>G **big dog eats**
* Given naive intermediate code for a C loop, hand-optimize the code, reducing by at least 20% the number of intermediate code instructions needed while maintaining semantic correctness of the program.
* Define "register assignment" in the context of a compiler. Explain why register assignment is an important compiler optimization.
  + Register assignment is the act of assigning a large number of target program variables onto a small number of CPU registers.
  + Multiple variables can be given the same register, as they aren’t all going to be in use at the same time.
  + Two variables in use at the same time cannot be assigned to the same register without screwing things up.
  + Variables that cannot be assigned to a register are kept in RAM and loaded in/out for every read/write. This is much slower, so compilers must try to assign as many variables to registers as possible.