Overview Continued

**What design pattern does gcc use?**

**What are the steps used to complete the front-end portion of the compilation process?**

**Given regex b\*(abb\*)\*(a|e), what set of strings does it specify?**

**Given the strings of a’s and b’s containing consecutive a’s, what regular expression can be used to specify them?**

**What are the 2 disambiguation rules for regular expressions for tokens?**

**What are the 2 requirements for a scanner?**

**How do we determine an optimal solution for token matching?**

**What are the advantages and disadvantages of using English to specify tokens?**

**What are the advantages and disadvantages of using regular expressions for specifying tokens?**

**What are the advantages and disadvantages of using DFAs for specifying tokens?**

**What are the advantages and disadvantages of using NFAs for specifying tokens?**

Façade.

Scanning/lexical analysis, parsing/syntactic analysis, semantic analysis, intermediate code generation.

No consecutive a’s.

(a|b)\*aa(a|b)\*

Longest match and rule-order priority (earlier token class wins in length tie).

Mutually exclusive and exhaustive.

Get regexs, convert them into NFA, convert NFA to DFA, optimize/minimize DFA.

It’s understandable but it’s tedious, imprecise, and incomplete.

It’s compact, precise, and complete but not operational.

Operational (can follow step by step) but not easy to build a combined DFA.

Easy to build from regexes and convert to DFA but is non-deterministic.