Several techniques for implementing Prolog in a efficient manner have been devised since the original interpreter, many of them aimed at achieving more speed. There are two main approaches to efficient Prolog implementation: (1) compilers to bytecode and then interpreting it (emulators) or (2) compilers to native code. Emulators have smaller loadcompilation time and are a good solution for their simplicity when speed is not a priority. Compilers are more complex than emulators, and the difference is much more acute if some form of code analysis is performed as part of the compilation, which impacts development time. Generation of low level code promises faster programs at the expense of using more resources during the compilation phase. In our work besides using an mixed execution mode, we design an optimizing compiler that using type feedback profiling, dynamic compilation and dynamic deoptimization for improving the performance of logic programming languages.