PARROT - INSTRUCTIONS FORMAT

Parrot can currently accept instructions to execute in four forms. PIR *ParrotIntermediateRepresentation* is designed to be written by people and generated by compilers. It hides away some low-level details, such as the way parameters are passed to functions.

PASM *ParrotAssembly* is a level below PIR - it is still human readable/writable and can be generated by a compiler, but the author has to take care of details such as calling conventions and register allocation. PAST *ParrotAbstractSyntaxTree* enables Parrot to accept an abstract syntax tree style input - useful for those writing compilers.

All of the above forms of input are automatically converted inside Parrot to PBC *ParrotBytecode*. This is much like machine code, but understood by the Parrot interpreter.

It is not intended to be human-readable or human-writable, but unlike the other forms execution can start immediately without the need for an assembly phase. Parrot bytecode is platform independent.

Instruction set

The Parrot instruction set includes arithmetic and logical operators, compare and branch/jump *forimplementingloops*, *if. . . thenconstructs*, *etc.* , finding and storing global and lexical variables, working with classes and objects, calling subroutines and methods along with their parameters, I/O, threads and more.