BCI Bytecode

Contents

Assembly and Bytecode

Unlike machine code (and other bytecode) BCI bytecode has dynamic opcodes. This means that bytecode is **not** necessarily portable.

This makes sense since the BCI instruction set can be extended for applications.

If one wants to share code that should run on any BCI it should be shared as assembly. The assembler will then use the local interpreter definition and generate suiting bytecode.

The Dynamic Instruction Set

The BCI comes with a set of prepared instructions. These are complete and provide a way to do basic operations like routines, loops and branching.

The methods are organized in a binary tree internally. To build the tree in a comfortable way there is an autoinserter that can insert up to `1023` methods into the tree.

The autoinserter creates the opcode basing on the order of the method that he inserts.

Byte Code Interpreter Definition

A Bytecode Interpreter Definition consists of two mayor parts: The memory definition that defines the number of data registers (up to 63), the number of memory words (up to 65535) and the number of program memory words (up to 65535).

The second part defines the commands. The definition contains bot the order of the commands (see The Dynamic Instruction Set) and the required arguments.