Disclaimer

This document is not comprehensive of all Golang features.

Contents

1 .gobc File Format

This document describes the Golang bytecode file format, .gobc. Each .gobc file contains the bytecode for a Go source file.

A class file consists of a stream of 8-bit bytes. All 16-bit, 32-bit, and 64-bit quantities are constructed by reading in two, four, and eight consecutive 8-bit bytes, respectively.

This document uses a short-hand for specifying the number of bytes associated with the data. The types u8, u16, u32, and u64 represent a one-, two-, four- or eight-byte quantity, respectively.

This document presents the .gobc file format using pseudo-code of C syntax. Arrays are zero-indexed.

1.1 gobcFile structure

A .gobc file consists of a single gobcFile structure:

```
gobcFile {
        u32           magicNumber
        u64           functionCount
        functionInfo  functions[functionCount]
}
```

The items that appear in the gobcFile structure are defined below:

magicNumber

The magicNumber item supplies the magic number identifying the gobc file format; it has the value 0xCAFEDEAD.

functionCount

The functionCount item specifies the number of functions defined in the file in the global scope.

functions

The **functions** item is an array where each element is a **functionInfo** structure giving a complete description of the function.

1.2 functionInfo structure

Each Golang function is described by a **functionInfo** structure.

TODO: How to handle anonymous and first-class functions?

```
functionInfo {
        u8              accessFlags
        u64             nameLength
        u8              name[nameLength]
        u32             attributesCount
        attributesInfo  attributes[attributesCount]
}
```

The items that appear in the **functionInfo** structure are defined below:

accessFlags

The value of the accessFlags item is a mask of flags used to denote access permission to and properties of this function. The interpretation of each flag, when set, is shown below.

Flag	Value	Description
EXPORTED	0x0001	Exported function; can be accessed outside the package

nameLength

The **nameLength** item specifies the number of bytes that comprise the function name.

name

The **name** item is an array of bytes that comprise the function name.

attributesCount

The attributesCount item specifies the number of attributes of this function.

attributes

The attributes item is an array where each element is an attributesInfo structure specifying an attribute.

The functionInfo structure can contain the following attribute structures:

- Code

1.3 attributeInfo structures

Attributes are attached to various structures to provide more detailed information. All attributes share the following general structure:

```
attributeInfo {
        u8          attributeType
        u64         attributeLength
        u1          data[attributesLength]
}
```

The items that appear in the attributeInfo structure are defined below:

attributeType

The attributeType item specifies which attribute the structure represents.

attributeLength

The attributeLength item specifies the length of the subsequent information in bytes. The length does not include the initial nine bytes that contain the attributeType and attributeLength items.

data

The data item contains the data of the attribute. Each attribute will have a different structure for this data.

1.3.1 Code attribute structure

The Code attribute is a variable-length attribute in the attributes table of a functionInfo structure. A Code attribute contains the bytecode instructions and auxiliary information for a single function.