NAME

llvm−bcanalyzer − LLVM bitcode analyzer

SYNOPSIS

llvm-bcanalyzer [*options*] [*filename*]

DESCRIPTION

The **llvm-bcanalyzer** command is a small utility for analyzing bitcode files. The tool reads a bitcode file (such as generated with the **llvm-as** tool) and produces a statistical report on the contents of the bitcode file. The tool can also dump a low level but human readable version of the bitcode file. This tool is probably not of much interest or utility except for those working directly with the bitcode file format. Most LLVM users can just ignore this tool.

If *filename* is omitted or is −, then **llvm-bcanalyzer** reads its input from standard input. This is useful for combining the tool into a pipeline. Output is written to the standard output.

OPTIONS

−nodetails
Causes **llvm-bcanalyzer** to abbreviate its output by writing out only a module level summary. The details for individual functions are not displayed.

−dump
Causes **llvm-bcanalyzer** to dump the bitcode in a human readable format. This format is significantly different from LLVM assembly and provides details about the encoding of the bitcode file.

−verify
Causes **llvm-bcanalyzer** to verify the module produced by reading the bitcode. This ensures that the statistics generated are based on a consistent module.

−help
Print a summary of command line options.

EXIT STATUS

If **llvm-bcanalyzer** succeeds, it will exit with 0. Otherwise, if an error occurs, it will exit with a non-zero value, usually 1.

SUMMARY OUTPUT DEFINITIONS

The following items are always printed by llvm−bcanalyzer. They comprize the summary output.

Bitcode Analysis Of Module
This just provides the name of the module for which bitcode analysis is being generated.

Bitcode Version Number
The bitcode version (not LLVM version) of the file read by the analyzer.

File Size
The size, in bytes, of the entire bitcode file.

Module Bytes
The size, in bytes, of the module block. Percentage is relative to File Size.

Function Bytes
The size, in bytes, of all the function blocks. Percentage is relative to File Size.

Global Types Bytes
The size, in bytes, of the Global Types Pool. Percentage is relative to File Size. This is the size of the definitions of all types in the bitcode file.

Constant Pool Bytes
The size, in bytes, of the Constant Pool Blocks Percentage is relative to File Size.

Module Globals Bytes
Ths size, in bytes, of the Global Variable Definitions and their initializers. Percentage is relative to File Size.

Instruction List Bytes
The size, in bytes, of all the instruction lists in all the functions. Percentage is relative to File Size. Note that this value is also included in the Function Bytes.

Compaction Table Bytes
The size, in bytes, of all the compaction tables in all the functions. Percentage is relative to File Size. Note that this value is also included in the Function Bytes.

Symbol Table Bytes
The size, in bytes, of all the symbol tables in all the functions. Percentage is relative to File Size. Note that this value is also included in the Function Bytes.

Dependent Libraries Bytes
The size, in bytes, of the list of dependent libraries in the module. Percentage is relative to File Size. Note that this value is also included in the Module Global Bytes.

Number Of Bitcode Blocks
The total number of blocks of any kind in the bitcode file.

Number Of Functions
The total number of function definitions in the bitcode file.

Number Of Types
The total number of types defined in the Global Types Pool.

Number Of Constants
The total number of constants (of any type) defined in the Constant Pool.

Number Of Basic Blocks
The total number of basic blocks defined in all functions in the bitcode file.

Number Of Instructions
The total number of instructions defined in all functions in the bitcode file.

Number Of Long Instructions
The total number of long instructions defined in all functions in the bitcode file. Long instructions are those taking greater than 4 bytes. Typically long instructions are GetElementPtr with several indices, PHI nodes, and calls to functions with large numbers of arguments.

Number Of Operands
The total number of operands used in all instructions in the bitcode file.

Number Of Compaction Tables
The total number of compaction tables in all functions in the bitcode file.

Number Of Symbol Tables
The total number of symbol tables in all functions in the bitcode file.

Number Of Dependent Libs
The total number of dependent libraries found in the bitcode file.

Total Instruction Size
The total size of the instructions in all functions in the bitcode file.

Average Instruction Size
The average number of bytes per instruction across all functions in the bitcode file. This value is computed by dividing Total Instruction Size by Number Of Instructions.

Maximum Type Slot Number
The maximum value used for a type's slot number. Larger slot number values take more bytes to encode.

Maximum Value Slot Number
The maximum value used for a value's slot number. Larger slot number values take more bytes to encode.

Bytes Per Value
The average size of a Value definition (of any type). This is computed by dividing File Size by the total number of values of any type.

Bytes Per Global
The average size of a global definition (constants and global variables).

Bytes Per Function
The average number of bytes per function definition. This is computed by dividing Function Bytes by Number Of Functions.

# of VBR 32−bit Integers
The total number of 32−bit integers encoded using the Variable Bit Rate encoding scheme.

# of VBR 64−bit Integers
The total number of 64−bit integers encoded using the Variable Bit Rate encoding scheme.

# of VBR Compressed Bytes
The total number of bytes consumed by the 32−bit and 64−bit integers that use the Variable Bit Rate encoding scheme.

# of VBR Expanded Bytes
The total number of bytes that would have been consumed by the 32−bit and 64−bit integers had they not been compressed with the Variable Bit Rage encoding scheme.

Bytes Saved With VBR
The total number of bytes saved by using the Variable Bit Rate encoding scheme. The percentage is relative to # of VBR Expanded Bytes.

DETAILED OUTPUT DEFINITIONS

The following definitions occur only if the −nodetails option was not given. The detailed output provides additional information on a per-function basis.

Type
The type signature of the function.

Byte Size
The total number of bytes in the function's block.

Basic Blocks
The number of basic blocks defined by the function.

Instructions
The number of instructions defined by the function.

Long Instructions
The number of instructions using the long instruction format in the function.

Operands
The number of operands used by all instructions in the function.

Instruction Size
The number of bytes consumed by instructions in the function.

Average Instruction Size
The average number of bytes consumed by the instructions in the function. This value is computed by dividing Instruction Size by Instructions.

Bytes Per Instruction
The average number of bytes used by the function per instruction. This value is computed by dividing Byte Size by Instructions. Note that this is not the same as Average Instruction Size. It computes a number relative to the total function size not just the size of the instruction list.

Number of VBR 32−bit Integers
The total number of 32−bit integers found in this function (for any use).

Number of VBR 64−bit Integers
The total number of 64−bit integers found in this function (for any use).

Number of VBR Compressed Bytes
The total number of bytes in this function consumed by the 32−bit and 64−bit integers that use the Variable Bit Rate encoding scheme.

Number of VBR Expanded Bytes
The total number of bytes in this function that would have been consumed by the 32−bit and 64−bit integers had they not been compressed with the Variable Bit Rate encoding scheme.

Bytes Saved With VBR
The total number of bytes saved in this function by using the Variable Bit Rate encoding scheme. The percentage is relative to # of VBR Expanded Bytes.

SEE ALSO

llvm-dis, <http://llvm.org/docs/BitCodeFormat.html>

AUTHORS

Maintained by the LLVM Team (<http://llvm.org/>).