NAME

llc - LLVM static compiler

SYNOPSIS

llc [*options*] [*filename*]

DESCRIPTION

The **llc** command compiles LLVM source inputs into assembly language for a specified architecture. The assembly language output can then be passed through a native assembler and linker to generate a native executable.

The choice of architecture for the output assembly code is automatically determined from the input file, unless the *-march* option is used to override the default.

OPTIONS

If **filename** is "**-**" or omitted, **llc** reads from standard input. Otherwise, it will read from **filename**. Inputs can be in either the LLVM assembly language format (**.ll**) or the LLVM bitcode format (**.bc**).

If the *-o* option is omitted, then **llc** will send its output to standard output if the input is from standard input. If the *-o* option specifies "**-**", then the output will also be sent to standard output.

If no *-o* option is specified and an input file other than "**-**" is specified, then **llc** creates the output filename by taking the input filename, removing any existing **.bc** extension, and adding a **.s** suffix.

Other **llc** options are described below.

End-user Options

-help
Print a summary of command line options.

-o <filename>
Use **<filename>** as the output filename. See the summary above for more details.

-O=uint
Generate code at different optimization levels. These correspond to the **-O0**, **-O1**, **-O2**, and **-O3** optimization levels used by **clang**.

-mtriple=<target triple>
Override the target triple specified in the input file with the specified string.

-march=<arch>

Specify the architecture for which to generate assembly, overriding the target encoded in the input file. See the output of **llc -help** for a list of valid architectures. By default this is inferred from the target triple or autodetected to the current architecture.

-mcpu=<cpuname>

Specify a specific chip in the current architecture to generate code for. By default this is inferred from the target triple and autodetected to the current architecture. For a list of available CPUs, use:

```
llvm-as < /dev/null | llc -march=xyz -mcpu=help
```

-filetype=<output file type>

Specify what kind of output **llc** should generated. Options are: **asm** for textual assembly (**'.s'**), **obj** for native object files (**'.o'**) and **null** for not emitting anything (for performance testing).

Note that not all targets support all options.

-mattr=a1,+a2,-a3,...

Override or control specific attributes of the target, such as whether SIMD operations are enabled or not. The default set of attributes is set by the current CPU. For a list of available attributes, use:

```
llvm-as < /dev/null | llc -march=xyz -mattr=help
```

--frame-pointer

Specify effect of frame pointer elimination optimization (all,non-leaf,none).

--disable-excess-fp-precision

Disable optimizations that may produce excess precision for floating point. Note that this option can dramatically slow down code on some systems (e.g. X86).

--enable-no-infs-fp-math

Enable optimizations that assume no Inf values.

--enable-no-nans-fp-math

Enable optimizations that assume no NAN values.

--enable-no-signed-zeros-fp-math

Enable FP math optimizations that assume the sign of 0 is insignificant.

--enable-no-trapping-fp-math
Enable setting the FP exceptions build attribute not to use exceptions.

--enable-unsafe-fp-math
Enable optimizations that make unsafe assumptions about IEEE math (e.g. that addition is associative) or may not work for all input ranges. These optimizations allow the code generator to make use of some instructions which would otherwise not be usable (such as **fsin** on X86).

--stats
Print statistics recorded by code-generation passes.

--time-passes
Record the amount of time needed for each pass and print a report to standard error.

--load=<dso_path>
Dynamically load **dso_path** (a path to a dynamically shared object) that implements an LLVM target. This will permit the target name to be used with the *-march* option so that code can be generated for that target.

-meabi=[default|gnu|4|5]
Specify which EABI version should conform to. Valid EABI versions are *gnu*, *4* and *5*. Default value (*default*) depends on the triple.

-stack-size-section
Emit the .stack_sizes section which contains stack size metadata. The section contains an array of pairs of function symbol values (pointer size) and stack sizes (unsigned LEB128). The stack size values only include the space allocated in the function prologue. Functions with dynamic stack allocations are not included.

-remarks-section
Emit the __remarks (MachO) section which contains metadata about remark diagnostics.

Tuning/Configuration Options

--print-after-isel
Print generated machine code after instruction selection (useful for debugging).

--regalloc=<allocator>
Specify the register allocator to use. Valid register allocators are:

basic

Basic register allocator.

fast

Fast register allocator. It is the default for unoptimized code.

greedy

Greedy register allocator. It is the default for optimized code.

pbqp

Register allocator based on 'Partitioned Boolean Quadratic Programming'.

--spiller=<spiller>

Specify the spiller to use for register allocators that support it. Currently this option is used only by the linear scan register allocator. The default **spiller** is *local*. Valid spillers are:

simple

Simple spiller

local

Local spiller

Intel IA-32-specific Options

--x86-asm-syntax=[att|intel]

Specify whether to emit assembly code in AT&T syntax (the default) or Intel syntax.

EXIT STATUS

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

SEE ALSO

lli(1)

AUTHOR

Maintained by the LLVM Team (https://llvm.org/).

COPYRIGHT