NAME

llvm-objdump - LLVM's object file dumper

SYNOPSIS

llvm-objdump [*commands*] [*options*] [*filenames...*]

DESCRIPTION

The **llvm-objdump** utility prints the contents of object files and final linked images named on the command line. If no file name is specified, **llvm-objdump** will attempt to read from *a.out*. If - is used as a file name, **llvm-objdump** will process a file on its standard input stream.

COMMANDS

At least one of the following commands are required, and some commands can be combined with other commands:

-a, --archive-headers
Display the information contained within an archive's headers.

-d, --disassemble
Disassemble all executable sections found in the input files. On some architectures (AArch64, PPC64, x86), all known instructions are disassembled by default. On the others, *--mcpu* or *--mattr* is needed to enable some instruction sets. Disabled instructions are displayed as **<unknown>**.

-D, --disassemble-all
Disassemble all sections found in the input files.

--disassemble-symbols=<symbol1[,symbol2,...]>
Disassemble only the specified symbols. Takes demangled symbol names when *--demangle* is specified, otherwise takes mangled symbol names. Implies *--disassemble*.

--dwarf=<value>
Dump the specified DWARF debug sections. The supported values are:

frames - .debug_frame

-f, --file-headers
Display the contents of the overall file header.

--fault-map-section
Display the content of the fault map section.

-h, --headers, --section-headers
Display summaries of the headers for each section.

--help
Display usage information and exit. Does not stack with other commands.

-p, --private-headers
Display format-specific file headers.

-r, --reloc
Display the relocation entries in the file.

-R, --dynamic-reloc
Display the dynamic relocation entries in the file.

--raw-clang-ast
Dump the raw binary contents of the clang AST section.

-s, --full-contents
Display the contents of each section.

-t, --syms
Display the symbol table.

-T, --dynamic-syms
Display the contents of the dynamic symbol table.

-u, --unwind-info
Display the unwind info of the input(s).

This operation is only currently supported for COFF and Mach-O object files.

-v, --version
Display the version of the **llvm-objdump** executable. Does not stack with other commands.

-x, --all-headers
Display all available header information. Equivalent to specifying *--archive-headers*, *--file-headers*, *--private-headers*, *--reloc*, *--section-headers*, and *--syms*.

OPTIONS

llvm-objdump supports the following options:

--adjust-vma=<offset>
Increase the displayed address in disassembly or section header printing by the specified offset.

--arch-name=<string>
Specify the target architecture when disassembling. Use *--version* for a list of available targets.

--build-id=<string>
Look up the object using the given build ID, specified as a hexadecimal string. The found object is handled as if it were an input filename.

-C, --demangle
Demangle symbol names in the output.

--debug-file-directory <path>
Provide a path to a directory with a *.build-id* subdirectory to search for debug information for stripped binaries. Multiple instances of this argument are searched in the order given.

--debuginfod, --no-debuginfod
Whether or not to try debuginfod lookups for debug binaries. Unless specified, debuginfod is only enabled if libcurl was compiled in (**LLVM_ENABLE_CURL**) and at least one server URL was provided by the environment variable **DEBUGINFOD_URLS**.

--debug-vars=<format>
Print the locations (in registers or memory) of source-level variables alongside disassembly. **format** may be **unicode** or **ascii**, defaulting to **unicode** if omitted.

--debug-vars-indent=<width>
Distance to indent the source-level variable display, relative to the start of the disassembly. Defaults to 52 characters.

-j, --section=<section1[,section2,...]>
Perform commands on the specified sections only. For Mach-O use *segment,section* to specify the section name.

-l, --line-numbers
When disassembling, display source line numbers. Implies *--disassemble*.

-M, --disassembler-options=<opt1[,opt2,...]>

Pass target-specific disassembler options. Available options:

- **reg-names-std**: ARM only (default). Print in ARM ’s instruction set documentation, with r13/r14/r15 replaced by sp/lr/pc.
- **reg-names-raw**: ARM only. Use r followed by the register number.
- **no-aliases**: AArch64 and RISC-V only. Print raw instruction mnemonic instead of pseudo instruction mnemonic.
- **numeric**: RISC-V only. Print raw register names instead of ABI mnemonic. (e.g. print x1 instead of ra)
- **att**: x86 only (default). Print in the AT&T syntax.
- **intel**: x86 only. Print in the intel syntax.

--mcpu=<cpu-name>
Target a specific CPU type for disassembly. Specify **--mcpu=help** to display available CPUs.

--mattr=<a1,+a2,-a3,...>
Enable/disable target-specific attributes. Specify **--mattr=help** to display the available attributes.

--no-leading-addr, --no-addresses
When disassembling, do not print leading addresses for instructions or inline relocations.

--no-print-imm-hex
Do not use hex format for immediate values in disassembly output.

--no-show-raw-insn
When disassembling, do not print the raw bytes of each instruction.

--offloading
Display the content of the LLVM offloading section.

--prefix=<prefix>
When disassembling with the *--source* option, prepend **prefix** to absolute paths.

--prefix-strip=<level>
When disassembling with the *--source* option, strip out **level** initial directories from absolute paths.

This option has no effect without *--prefix*.

--print-imm-hex
Use hex format when printing immediate values in disassembly output (default).

-S, --source
When disassembling, display source interleaved with the disassembly. Implies *--disassemble*.

--show-all-symbols
Show all symbols during disassembly, even if multiple symbols are defined at the same location.

--show-lma
Display the LMA column when dumping ELF section headers. Defaults to off unless any section has different VMA and LMAs.

--start-address=<address>
When disassembling, only disassemble from the specified address.

When printing relocations, only print the relocations patching offsets from at least **address**.

When printing symbols, only print symbols with a value of at least **address**.

--stop-address=<address>
When disassembling, only disassemble up to, but not including the specified address.

When printing relocations, only print the relocations patching offsets up to **address**.

When printing symbols, only print symbols with a value up to **address**.

--symbolize-operands
When disassembling, symbolize a branch target operand to print a label instead of a real address.

When printing a PC-relative global symbol reference, print it as an offset from the leading symbol.

When a bb-address-map section is present (i.e., the object file is built with **-fbasic-block-sections=labels**), labels are retrieved from that section instead.

Only works with PowerPC objects or X86 linked images.

Example:

A non-symbolized branch instruction with a local target and pc-relative memory access like

```
cmp eax, dword ptr [rip + 4112]
jge 0x20117e <_start+0x25>
```

might become

```
<L0>:
  cmp eax, dword ptr <g>
  jge     <L0>
```

--triple=<string>
Target triple to disassemble for, see **--version** for available targets.

-w, --wide
Ignored for compatibility with GNU objdump.

--x86-asm-syntax=<style>
Deprecated. When used with *--disassemble*, choose style of code to emit from X86 backend. Supported values are:

att AT&T-style assembly

intel
Intel-style assembly

The default disassembly style is **att**.

-z, --disassemble-zeroes
Do not skip blocks of zeroes when disassembling.

@<FILE>
Read command-line options and commands from response file <*FILE*>.

MACH-O ONLY OPTIONS AND COMMANDS

--arch=<architecture>
Specify the architecture to disassemble. see **--version** for available architectures.

--archive-member-offsets
Print the offset to each archive member for Mach-O archives (requires *--archive-headers*).

--bind
Display binding info

--data-in-code
Display the data in code table.

--dis-symname=<name>
Disassemble just the specified symbol's instructions.

--chained-fixups
Print chained fixup information.

--dyld-info
Print bind and rebase information used by dyld to resolve external references in a final linked binary.

--dylibs-used
Display the shared libraries used for linked files.

--dsym=<string>
Use .dSYM file for debug info.

--dylib-id
Display the shared library's ID for dylib files.

--exports-trie
Display exported symbols.

--function-starts [=<addrs|names|both>]
Print the function starts table for Mach-O objects. Either **addrs** (default) to print only the addresses of functions, **names** to print only the names of the functions (when available), or **both** to print the names beside the addresses.

-g Print line information from debug info if available.

--full-leading-addr
Print the full leading address when disassembling.

--indirect-symbols
Display the indirect symbol table.

--info-plist
Display the info plist section as strings.

--lazy-bind
Display lazy binding info.

--link-opt-hints
Display the linker optimization hints.

-m, --macho
Use Mach-O specific object file parser. Commands and other options may behave differently when used with **--macho**.

--no-leading-headers
Do not print any leading headers.

--no-symbolic-operands
Do not print symbolic operands when disassembling.

--non-verbose
Display the information for Mach-O objects in non-verbose or numeric form.

--objc-meta-data
Display the Objective-C runtime meta data.

--private-header
Display only the first format specific file header.

--rebase
Display rebasing information.

--rpaths
Display runtime search paths for the binary.

--universal-headers
Display universal headers.

--weak-bind
Display weak binding information.

XCOFF ONLY OPTIONS AND COMMANDS

--symbol-description
Add symbol description to disassembly output.

BUGS

To report bugs, please visit <*https://github.com/llvm/llvm-project/labels/tools:llvm-objdump/*>.

SEE ALSO

llvm-nm(1), **llvm-otool(1)**, **llvm-readelf(1)**, **llvm-readobj(1)**

AUTHOR

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

COPYRIGHT