Z80ASM

Updated: May 10, 2005

NAME

z80asm - assembler for the Z80 microprocessor

SYNOPSIS

z80asm [*options*] [*files...*]

DESCRIPTION

Z80asm is an assembler for Z80 assembly. If no input files are specified, stdin is used. If no output file is specified, "a.bin" is used. If "-" is specified as output file, stdout is used. This makes it possible to use the assembler in a pipeline.

When multiple input files are specified, the assembler first uses all files which were specified with -i or --input, in the order given. After that, all files which were specified as non-option arguments are assembled, also in the order given.

OPTIONS

-h, --help
: Show summary of options and exit.

-V, --version
: Display version information and exit.

-v, --verbose
: Be verbose. Specify multiple times to be more verbose. Messages are sent to standard error.

-l, --list[=filename]
: Write a list file. No filename or '-' means stderr.

-L, --label[=filename]
: Write a label file. No filename or '-' means stderr.

-p, --label-prefix=prefix
: prefix all labels with this prefix.

-i, --input=filename
: Specify an input file (-i may be omitted). '-' means stdin.

-o, --output=filename
: Specify the output file. '-' or completely omitting the option means stdout.

-I, --includepath=dirname
: Add a directory to the include path. The order in which the directories are tried is from back to front: the last directory specified has the highest priority. "/usr/share/z80asm" is always in the include path (with lowest priority), you don't have to specify it.

-f, --force
: Produce output even in case of errors. Normally the output, list and label files are removed when assembly is unsuccesful.

ASSEMBLER DIRECTIVES

All mnemonics and registers are case insensitive. All other text (in particular, labels and macros) are not. Undocumented opcodes are as much as possible supported:
sll and sli are equal and can both be used.
ixh, ixl, iyh and iyl can be used.

Assembler directives are:

incbin 'filename'

Include a binary file into the resulting assembled file. This can be used to include text files, or images, sound files, etc. The filename is searched for in the current directory, and then in the include path, just like for include. Also like for include, the quotes can be any character (but must match) and no substitution is performed (so ~ is not your home directory).

defb or **db** arg, arg, arg, ...

Define bytes.

defm or **dm** String, 'String'

Define message. Each character in the string is stored as one byte. Backslash escapes are allowed, as in characters in expressions. Unlike the argument for include, the quotes must really be quotes (but they can be single or double quotes. The closing quote must match the opening quote.)

defb/db and defm/dm are really aliases; either can take both quoted strings and numbers:

defb "This text should be in a buffer\r\n", 0

defs or **ds** count [, value]

Define space. count bytes are reserved. Each of them is initialised to the specified value, or 0 if no value is specified.

defw or **dw** arg, arg, arg, ...

Define words. Each argument is stored as two bytes, the low order byte first.

end

End assembly of this source file. Any remaining lines are copied into the list file (if present), but not assembled.

label: **equ** expression

Define label to have value expression.

if expression
code block 1
else
code block 2
else
code block 3
...
code block n
endif

Conditionally assemble code. If expression is not 0, all odd code blocks are assembled, if expression is 0, all even blocks are assembled. Usually only one or two code blocks are present.

include 'file'

Include file into the source. The quotes around the file for include are mandatory, but you can choose the quotes yourself. eg, you may use % or even a letter as a quote. The filename does not undergo any expansion, so \, ~, $, etc are passed as written (which means ~ will not be your home directory.) The filename is used as specified, and then prefixed with each directory in the include path, until it can be opened.

label: **macro** arg1, arg2, ...
code block
endif

Define a macro. The macro can be used where an opcode is expected. The code block is then substituted, with the given values for the arguments. This is a textual substitution, so the following example is valid:

makelabel name
label_name:
endm

This will generate a label with a constructed name (it's not a very useful example, but it shows the possiblities).

org address

Set the "program counter" to address. This does not add any bytes to the resulting binary, it only determines how the rest of the code is interpreted (in particular, the value of labels and **$**).

seek offset

Seek to position offset in the output file. This can be used for overwiting previously assembled code, for example for patching a binary which was included using **incbin**.

EXPRESSIONS

All expressions can use the following operators, in order of precedence: (**a**, **b** and **c** denote subexpressions)

a ? b : c

If a is not zero, return b, otherwise c

a | b

bitwise or

a ^ b

bitwise xor

a & b

bitwise and

a == b, a = b, a != b

equality

a <= b, a >= b, a < b, a > b

inequality

a << b, a >> b

bit shift

a + b, a - b

addition and subtraction

a * b, a / b, a % b

multiplication, division and modulo

~a, +a, -a

bitwise not, no effect and negation

?label

1 if label exists, 0 if it does not. This does not generate an error if label does not exist. Note that this is usually evaluated immediately (if the rest of the expression permits), and it does not check if the label is defined later. This means it can be used as the argument of **if** , to get the functionality of #ifdef in C.

(a)

Parenthesis. Literals in expressions may be written as: (case does not matter)

@c11

arbitrary base number (specified by 'c' so c+1 == 10: here base is 13)

14, 14d, @914

decimal number

016, 16o, 16q, &o16, @716

octal number

0Eh, 0xE, &hE, $E, @FE

hexadecimal number (for the first notations, the first character must be 0-9)

%1110, 1110b, &b1110, @11110
: binary number

's'
: ASCII code of 's'

'\\n', '\\r', '\\a', '\\t'
: Newline, carriage return, alert, tab

'\\nnn'
: Octal ASCII code

$
: address of first byte of current command

LABELS

In all expressions, labels may be used. However, there are some expressions where the value must be computable at once, and therefore only previously defined labels may be used. This is the case for:
- The argument of org
- The argument of seek
- The argument of equ (eg, a label definition)
- The first argument of ds
- The argument of if

In all other expressions, labels which are defined later may be used.

Labels must consist of letters, digits, underscores and periods, and must not start with a digit. Labels are case sensitive.

Labels starting with a period (.) are **local** , which means their scope is only the current include file or macro definition (and files included/macros called from it). This is particularly useful for macros, to prevent duplicate definitions when using a macro more than once.

EXIT STATUS

If assembly was successful, no output is produced (except the result, and messages triggered by --verbose) and 0 is returned. At any error, there is output on the standard error and 1 is returned.

NOTES

Parts that are not assembled because of an if statement and macros which are defined but never used are only checked to have a correct command. The argument is not parsed. This means that if the file passes through the assembler with no warnings or errors, it may still not assemble correctly in a different setting (where the if's give different results).

BUGS

If you find a bug, or want to send comments, please use the web interface at http://savannah.nongnu.org/projects/z80asm/ or send an e-mail to wijnen@debian.org.

AUTHOR

Z80asm was written by Bas Wijnen <wijnen@debian.org>.
Some patches were provided by Jan Wilmans <jw@dds.nl>