Z80Asm.py - Z80 Cross Assembler for Windows and Linux

Introduction

Z80Asm.py is a new assembler for Z80 and Z180 code, written in Python 3, so that it will run on both Windows and Linux (also any other operating system with a reasonable Python 3 implementation). It can generate machine code as either binary images or Intel Hex format files for writing to ROM (EPROM, EEPROM, or Flash).

One of the features of this program is that it supports three different styles of coding, outlined below. It is also able to reformat source code from one style to another.

MA Style

This is the style of coding used by Martin Allcorn's BBC Basic assembler. The identifying features of this style are:

- Labels (call and jump destinations and data addresses) appear on a line of their own, prefixed by a stop ".".
- Opcodes start in column 1.

This mode has been tested by assembling the source of the CFX-II ROM and confirming that the resulting binary image is consistent with Martin's original.

The special macros needed to assemble the "System Diagnostic Cartridge" have not been implemented. It would not be difficult to add these but I would prefer to add a general macro capability.

M80 Style

This is not a complete emulation of the CP/M M80 assembler, but more a convenient designation of the style of coding. The identifying features of this style are:

- Labels generally appear on the same line as the instruction to which they refer, followed by a colon ":".
- Opcodes are generally indented.

Features of the M80 assembler that are **not** currently reproduced include:

- 8080 opcodes.
- Code and data segments (CSEG and DSEG).
- Macros (may be added at a future date).
- Object file output for L80.

ZASM Style

This is similar to M80 style, differing only in the interpretation of pseudo-ops.

Usage

```
usage: Z80Asm.py [-h] [-v] [-b [BINARY]] [-f FILL] [-x [HEX]] [-n] [-l [LIST]]
                 [-a] [-o [OUTPUT]] [-r {MA,M80,ZASM}] [-m] [-k [KEEP]] [-e]
                 [-c {Z80,Z180}] -s {MA,M80,ZASM} [-p] [-d]
                 source

Assemble Z80 code written in different styles

positional arguments:
  source                The Z80 source file

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -b [BINARY], --binary [BINARY]
                        Machine code in binary format
  -f FILL, --fill FILL  Fill byte for undefined addresses
  -x [HEX], --hex [HEX]
                        Machine code in Intel hex format
  -n, --number-build    Append build number to assembled file names
  -l [LIST], --list [LIST]
                        List file
  -a, --address         Show load address as well as relocation
  -o [OUTPUT], --output [OUTPUT]
                        Reformatted source file
  -r {MA,M80,ZASM}, --reformat {MA,M80,ZASM}
                        Style for reformatted source (default M80)
  -m, --modeline        Emacs modeline in reformatted source
  -k [KEEP], --keep [KEEP]
                        Keep pass 1 list file
  -e, --echo            Echo source to screen
  -c {Z80,Z180}, --cpu {Z80,Z180}
                        The processor type
  -s {MA,M80,ZASM}, --style {MA,M80,ZASM}
                        The style of the Z80 source
  -p, --permissive      Ignore some syntax errors
  -d, --debug           Show assembler debug info
```

Details

-b [BINARY], --binary [BINARY]

> Save the machine code as a binary image file. Optionally followed by the name of the file. If no file name is specified, defaults to the root of the source file name with a ".img" extension.

-f FILL, --fill FILL

> If the source leaves gaps in the generated machine code, then these gaps have to be filled using padding bytes in the image. This option specifies the byte to use. If the switch is omitted, then a default of 0xFF is used.

-x [HEX], --hex [HEX]

> Save the machine code as an Intel hex file. Optionally followed by the name of the file. If no file name is specified, defaults to the root of the source file name with a ".hex" extension. It is possible to save both binary and hex images.

-n, --number-build

Append "__Bnnn", where nnn is the build number (see below) to the binary and/or hex image file names (before the extension).

-l [LIST], --list [LIST]

Save a list file showing the source code, generated machine code and any error messages. If no file name is specified, defaults to the root of the source file name with a ".lst" extension.

-a, --address

For relocated code (see below) show both the location in the binary image, and the addresses at which the code will be executed. If omitted only the execution addresses are shown.

-o [OUTPUT], --output [OUTPUT]

Save a reformatted version of the input source code. If no file name is specified, defaults to the root of the source file name with an extension the lower case version of the reformatted style.

-r {MA,M80,ZASM}

The coding style to use for reformatted source code. If not specified, defaults to "M80".

-m, --modeline

Adds an Emacs style modeline to the top of the list file and reformatted source file to specify the tab spacing (8 spaces) to use when displaying / editing the file. Many other editors understand Emacs modelines.

-k [KEEP], --keep [KEEP]

Normally the list file generated during the first pass is overwritten by the second pass listing. For some errors it can be useful to compare the first and second pass listing. This option preserves the listing from the first pass. If no file name is specified, then it defaults to the root name of the source file, followed by "_p1.lst".

-e, --echo

Echo source lines to the screen as they are processed in order to display progress. Otherwise only error messages and their corresponding source lines are displayed.

-c {Z80,Z180}, --cpu {Z80,Z180}

Specifies the CPU to assemble for. Defaults to "Z80". Specifying "Z180" enables additional opcodes.

-s {MA,M80,ZASM}, --style {MA,M80,ZASM}

Specifies the style of the source code. One of these options must be selected.

-p, --permissive

Allow a slightly looser coding style. Features enabled by this option include:

- Label matching in expressions is case insensitive. By default labels have to match in case.
- Labels may be multiply defined, providing all definitions result in the same value. By default labels may only be defined in one location.

-d, --debug

Include information to assist debugging the expression evaluator in the list file. Probably only of interest to the author.

A number of the above options are followed by an optional file name. If the next parameter following one of these options does not begin with a dash "-", it will be taken as the file name. This could result in the source file name being mistaken for the option file name. For this reason it is recommended that the options are specified in the order given above, with the style option last.

Relocated Code

For code that is initially loaded in ROM, but is then either paged to a different address, or copied into RAM there are two relevant addresses:

Location counter:
Location of the code in ROM / image file.
Program counter:
Location where the code is executed and address for calls and jumps.

The three different styles of coding differ in the pseudo-ops used to specify these addresses.

MA Style

OFFSET [address]

Sets the program counter to the specified address, and remembers the offset between the program counter and location counter. If no address is specified, resets the program counter to the location counter, and sets the offset to zero.

ORG address

Sets the location counter to the specified address and the program counter to the location counter plus the previously remembered offset.

M80 Style

ORG address

Sets both the location and program counters to the specified address.

.PHASE address

> Sets just the program counter to the specified address.

.DEPHASE

> Resets the program counter to the location counter.

ZASM Style

LOAD address

> Sets the location counter to the specified address.

ORG [address]

> Sets the program counter to the specified address. If no address is specified resets the program counter to the location counter. (Note: The original ZASM does not support omitting the ORG address).

Pseudo-ops

The following pseudo-ops are recognised by Z80Asm (in addition to the location ops given in the previous section). Most of these pseudo-ops are recognised for all styles, irrespective of which style they originated from.

IF expression

> If the expression evaluates to non-zero, code to the corresponding ENDIF is emitted. If zero, following code is not emitted. The following code must still be syntactically correct to enable reformatting.

ENDIF

> Terminates a section of conditional code.

label[:] EQU expression

> Asigns the value of the expression to a label. The colon after the label is optional (ZASM requires it).

END

> Marks the end of a source file.

LIST
.LIST

> Turns on output to the list file (if enabled).

NOLIST
.XLIST

Turns off output to the list file.

NAME name

Specifies a name for the code. Ignored by Z80Asm.

INCLUDE filename

Includes the contents of the specified source file at this location. File name matching is case insensitive (even on Linux). For MA style any RISCOS file type (comma and number) at the end of the file name is ignored. For the other styles, if no extension is specified it is assumed to be the same as that for the original source file specified on the command line. If generating reformatted output, all the included files are in the single output file, multiple outputs are not generated. The include statement is converted to a comment in the reformatted output.

DATE

Include the current date as a 10 byte string in the generated machine code.

BUILD

If not previously used, read the current build number from the file "build" in the current directory (as a 4 byte integer), increment it and save the new value. Include the build number as a 5 byte string in the generated machine code.

EQUD decimal_number

Include the specified number as a 4 byte little-endian integer value in the generated machine code.

DB expression [,expression]...
DEFB expression [,expression]...

Include the specified list of byte values in the generated machine code. If the expression is a string, then all the bytes of the string are output.

DW expression [,expression]...
DEFW expression [,expression]...

Include the specified list of values in the generated machine code as 2 byte little-endian word values.

DC string [,string]...
DEFC string [,string]...

Include each of the listed strings in the generated machine code, with the msb of the last byte of each string set.

(M80 style only) DS value

> Reserve the specified number of bytes of space in the machine code for variables. In a binary image the space is padded with the fill byte.

(MA and ZASM styles) DS string, [,string]...
DEFS string, [,string]...

> Include each of the listed strings in the generated machine code.

BYTE expression

> Reserve the specified number of bytes of space in the machine code for variables. In a binary image the space is padded with the fill byte.

WORD expression

> Reserve the specified number of 2 byte words of space in the machine code for variables. In a binary image the space is padded with the fill byte.

Expressions

As far as practical Z80Asm allows expressions from any style to be used in any style of source code, only restricting items to a particular style where otherwise there would be ambiguity.

Numeric Constants

In the following, the number of digits shown (as lower case letters) is illustrative only, and the number used (up to the maximum allowed by the data type) is optional.

%bbbbbbbb
bbbbbbbbB

> Binary constant.

nnnnnnQ

> Octal constant.

dddd
ddddD

> Decimal constant.

&hhhh (MA style only)
#hhhh
dhhhH (the leading digit must be 0-9, so for example 0F123H)
0xhhhh

> Hex constant.

"c"
+ASC"c" (MA style only)

The ASCII value of a single character.

False is represented by zero, and true by all bits set.

String Constants

Strings may be enclosed in single or double quotes. The opening and closing quotes must match. A quote of the oposite style may be included. To include a quote of the enclosing style, double it. In ZASM style, bytes in a string may be specified in hex with the escape sequence \xx. A backslash is generated by doubling it.

Operators

The following operators are supported in expressions:

Operator	**Rank**	**Description**
Brackets	10	Sub-expression.
LOW	9	Low byte of value.
HIGH	9	High byte of value.
*	8	Multiply.
/	8	Divide.
MOD	8	Modulus.
SHL	8	Shift left.
SHR	8	Shift right.
+	7	Unitary plus.
-	7	Unitary minus.
~	7	Bitwise not.
+	6	Binary addition.
-	6	Binary subtraction.
LT	5	Less than.
LE	5	Less than or equal.
EQ	5	Equal.
NE	5	Not equal.
GE	5	Greater than or equal.
GT	5	Greater than.
NOT	4	Bitwise not.
&	3	Bitwise and.
AND	3	Bitwise and.
!	2	Bitwise or.
OR	2	Bitwise or.
^	2	Bitwise xor.
XOR	2	Bitwise xor.
End of expression	1	End of expression