Z80Asm —TRS-80 Assembler for Windows

Description

```
Usage:  z80asm [options]
Options:
-cmd: Output a TRSDOS CM
-cim: Output a core-imag
-nc: disable display of
-nh: disable generation
-nm: disable display of
-sl: suppress display of
```

Z80Asm is a TRS-80 assembler for the twenty-first century. It assembles TRS-80 assembly language programs and creates /CMD files that will run on a real or emulated TRS-80, but it runs on the command line of your Windows PC. That means you can use your favorite Windows text editor to create and edit your assembly language source files and that your source files will be stored on your Windows hard drive (and can easily be backed up onto CD's, DVD's, thumb drives, and so on). Z80Asm is the modern, easy way to do assembly language development for your TRS-80 Models I, III, and 4.

Note: Windows PC's and TRS-80's handle their filenames and extensions differently. On a Windows PC, the filename and extension are separated by a period (FILENAME.EXT); on the TRS-80 they are separated by a slash (FILENAME/EXT). Since most of the files handled by Z80Asm will actually reside on a Windows PC, this instruction manual uses the Windows style of naming (except in a few cases where the files reside on a TRS-80 virtual disk).

Table of contents:

Z80Asm —TRS-80 Assembler for Windows

Installation

The Z80Asm assembler is a single executable file that runs from the command line. It does not have (or need) an installation program; just copy it into the directory where you want to use it, or copy it into a binaries directory and use a PATH statement to make it visible to the system.

Usage

Invoking Z80Asm is simple and follows the conventions of most of the popular TRS-80 assemblers. The command line looks like this:

```
Z80ASM [options] file
```

Any options are (as the name implies) optional; if you don't supply any, Z80Asm will assemble the source file named FILE and will generate a TRS-80 program file named FILE.CMD. FILE.CMD can then be copied to a real or emulated TRS-80 for testing. By default, Z80Asm will also create the file named FILE.LST, which is a listing of the assembled program.

Please note that because different people use different extensions for their assembly language source files (ASM, SRC, etc.), Z80Asm does not add a default file extension. If you want to assemble FILE.ASM, you need to type FILE.ASM on the command line; if you want to assemble FILE.SRC, you need to type FILE.SRC.

This default behavior will probably be enough in most circumstances, but Z80Asm features the following list of options to serve your needs.

Assembly Options

- **-cmd**: The default behavior, to output a TRSDOS-style CMD file. (This option never actually needs to be specified because it is Z80Asm's default behavior, but it is included on this options list for completeness.)
- **-hex**: Output an Intel-style HEX file. If you don't know what a HEX file is, you don't need this feature, but if you do need to create Intel HEX files, Z80Asm can do it for you. (This setting is only available in the registered assembler.)
- **-cim**: Output a core image file. A core image file is just a straight "dump" of information; it doesn't contain all of the loading instructions and bookkeeping information that TRS-80 program files require. (This setting is only available in the registered assembler.)
- **-com**: Output a CP/M COM file. This option will help you use Z80Asm to develop CP/M programs for your TRS-80. (This setting is only available in the registered assembler.)
- **-nc**: Disable the display of false conditionals. Normally when you include IF/THEN/ELSE conditionals in your source code, Z80Asm will include both the true and the false parts of the conditionals in your program listing. You can keep your program listings shorter and neater by only including true conditionals.
- **-nh**: Disable the generation of a CMD header record. TRS-80 CMD files usually begin with an eight-byte header that looks like this: X'05 06 xx xx xx xx xx xx' (replace the six 'xx' characters with the first six bytes of the disk filename). Use this option if for some reason you don't want this header included in your disk file.
- **-nm**: Disable display of macro expansions. Z80Asm has very complete and robust macro capabilities, but if you use a lot of macros in your programs you probably don't want to see them expanded every time they appear in the program listing.
- **-sl**: Suppress display of local labels. As far as Z80Asm is concerned, local labels are labels that start with a dollar sign ("$"). If you use the dollar sign to start all your local labels and then specify the -sl option when you assemble your source code, local labels will not be included in the symbol table listing. This will make that listing much less cluttered and much easier to follow and understand.
- **-no**: Disable generation of an output file. You might just want to check your source file for errors, or generate a program listing but no program file.
- **-nl**: Disable generation of a listing file. Z80Asm creates a program listing by default, but you may not want or need one.
- **-z1**: Enable support of Hitachi HD64180 instructions. One of the more popular TRS-80 add-ons in its later years was the XLR8er board, which provided additional memory and a faster and more capable Hitachi HD64180 microprocessor. This option enables Z80Asm to generate the extra HD64180 instructions that are supported by XLR8-ed TRS-80's.
- **-equ[=limit]**: Enable generation of an equate listing. An equate listing is a list of all the equate values that were used in a source file; if you have ever looked at *The Source*, the published program listings for the TRSDOS 6 operating system, you know what one looks like. You can limit the equate listing to only equates with names that contain a special character using the =limit sub-option. For example, -equ=limit will generate an equate listing, but only of equates with names that contain non-alphanumeric characters. (This setting is only available in the registered assembler.)
- **-i:<dir>**: Set the include directory. By default Z80Asm looks for include files in the current directory. You may want it to look somewhere else to keep your subdirectory structure neater.
- **-iv:<file>**: Set a virtual disk as input. By default Z80Asm reads its files from the "normal" Windows hard drive, but you may want it to read its files from a TRS-80 virtual disk. Please note that the virtual disk

cannot currently be in use by another program — for example, mounted in a drive on a currently-running TRS-80 emulator. (This setting is only available in the registered assembler.)

- **-ov:<file>**: Set a virtual disk as output. By default Z80Asm writes its output file to the "normal" Windows hard drive, but you may want it to write it directly onto a TRS-80 virtual disk instead. Again please note that the virtual disk cannot currently be in use by another program. (This setting is only available in the registered assembler.)
- **-o:<file>**: Rename the output file. Z80Asm gives the output files the same name as the source file by default; in other words, PROGRAM.ASM generates PROGRAM.CMD and PROGRAM.LST. Use this option to change the name of the output files.

Here are some examples of possible Z80Asm command lines. To load in the SAMPLE.SRC source file, assemble it, and write out the SAMPLE.CMD program file and SAMPLE.LST program listing:

```
Z80ASM SAMPLE.SRC
```

To load and assemble the same source file, write out the SAMPLE.CMD program file, but skip the generation of the SAMPLE.LST program listing:

```
Z80ASM -nl SAMPLE.SRC
```

To load the PROGRAM/ASM source file from the TRS-80 virtual disk DSKIMAGE.DSK, assemble it, disable the display of macro expansions, and write the program file PROGRAM/CMD and program listing PROGRAM/LST to the same virtual disk:

```
Z80ASM -nm -iv:DSKIMAGE.DSK -ov:DSKIMAGE.DSK PROGRAM/ASM
```

To load the XLR8.ASM source file, assemble it with support for the Hitachi HD64180 instructions, generate an equate listing of non-alphanumeric equates, and write out the XLR8.CMD program file and XLR8.LST program listing:

```
Z80ASM -z1-equ=limit XLR8.ASM
```

Format for the Assembly Language Program

The format for the assembly language programs Z80Asm accepts follows the same rules as earlier TRS-80 (and most other) assemblers. If you are familiar with assemblers like MRAS, Zeus, or M-ZAL, or MASM or TASM, you will be completely at ease with Z80Asm and its requirements.

Each line of the program looks something like this, but keep in mind that each of the four fields is optional:

[LABEL]   [OPCODE]   [OPERANDS]   [; comment]

Each field is separated by at least one space or tab, but aside from that there are no restrictions about how a program is formatted. You can make each field line up in columns or not; your programming style is completely up to you. Z80Asm treats upper- and lower-case letters as identical (except inside character strings), so you can decide whether to program in upper or lower case.

Field 1: The label is a symbolic name for a memory address. The label can be from two to fifteen characters long. Labels may be terminated by a colon, a space, a tab, or any combination of the three. The first character of a label must be a letter, an underline, a dollar sign, or an at sign. The subsequent characters of a label may be any of these characters as well as a digit (0 through 9) or a question mark.

Fields 2 and 3: The opcode and operands are standard Z-80 assembly language opcodes and operands as described in the Series I Editor/Assembler manual and other similar publications.

Field 4: The comment always begins with a semi-colon and ends with the end of the line.

Expressions

As can most other assemblers, Z80Asm can evaluate expressions as part of the source code. The expressions can include simple arithmetic:

- Addition (the plus sign, as in **number1+number2**)
- Subtraction (the minus sign, as in **number1-number2**)
- Multiplication (the asterisk, as in **number1*number2**)
- Division (the slash, as in **number1/number2**)
- Modulo division (the word .MOD., as in **number1.MOD.number 2**)

… and logical comparisons:

- Logical (bitwise) AND (the word .AND. or an ampersand, as in **number1.AND.number2** or **number1&number2**)
- Logical (bitwise) OR (the word .OR. or an exclamation point, as in **number1.OR.number2** or **number1!number2**)
- Logical (bitwise) exclusive OR (the word .XOR., as in **number1.XOR.number2**)
- Logical one's complement (the word .NOT., as in **LABEL EQU .NOT.number1**)
- Logical (binary) not equal (the word .NE., as in **IF number1.NE.number2**)
    - Logical (binary) equal (the word .EQ., as in **IF number1.EQ.number2**)
    - Logical (binary) greater than (the word .GT., as in **IF number1.GT.number2**)
    - Logical (binary) greater than or equal (the word .GE., as in **IF number1.GE.number2**)
    - Logical (binary) less than (the word .LT., as in **IF number1.LT.number2**)
    - Logical (binary) less than or equal (the word .LE., as in **IF number1.LE.number2**)

… and bit manipulation:

- Shift left (less than sign, as in **number1<number2**)
- Shift right (less than sign and a negative number, as in **number1<-number2**)
- Shift number1 left (the word .SHL., as in **number1.SHL.number2**)
- Shift number1 right (the word .SHR., as in **number1.SHR.number2**)
- Get the high order byte (the word .HIGH., as in **.HIGH.number1**)
- Get the low order byte (the word .LOW., as in **.LOW.number1**)

Pseudo-Ops

Pseudo-ops look like Z-80 assembly language instructions but are actually commands to the assembler. Z80Asm handles all of the familiar pseudo-ops, including (but not limited to):

ORG specifies the location in memory where the program will start assembling. Example: **ORG 3000H**.

END marks the end of a program file and optionally specifies the location of the start of the program. Examples: **END** or **END START** or **END 3000H**.

EQU sets a label to a specified value. Examples: **LABEL1 EQU 3200H** or **LABEL2 EQU LABEL1+256**.

DEFL also sets a label to a specified value, but DEFL labels can be changed multiple times in the program.

DB inserts a single byte or a string of bytes or characters into the program. DEFB, DEFM, and DM are all synonyms of DB. Examples: **DB 13**, **DB ‘This is a message'**, **DB 1,2,3,4,5**.

DW and its synonym DEFW insert a word value or a series of word values into the program. Examples: **DW 1024** or **DW 1200H,0400H,5213H**.

DS and its synonym DEFS reserve blank, empty space in the program. Example: **DS 200H**.

DATE inserts the system date into your program as an eight-character string.

TIME inserts the system time into your program as an eight-character string.

DC inserts the specified number of byte constants into the program. Examples: **DC 64,'-‘** or **DC 25,0**.

IF/ELSE/ENDIF constructions are used to build conditional statements.

MACRO and ENDM are used to set up macro statements (see later in this document for more details).

Assembler Directives

Assembler directives are commands embedded in the program that tell the assembler to do something different. Z80Asm supports several directives:

*GET FILENAME or *INCLUDE FILENAME tell Z80Asm to start reading and assembling the contents of FILENAME as though it were included at the current position. When FILENAME reaches the end of the file or an END statement, assembly will resume in the original file where it left off. Include files are very useful for loading macro definitions or for breaking a larger program up into multiple files.

*LIST OFF stops the assembly listing (but not the assembly itself).

*LIST ON turns the assembly listing on again after a *LIST OFF.

*MOD changes the module character substitution string. This string is used in local labels to replace the question mark in labels like **$LOOP?**. If you are writing your program in modules and use the *MOD directive at the beginning of every module, you will be able to avoid the problem of duplicate local labels.

*RADIX changes the default base of all numeric terms in your program. Z80Asm normally uses a radix of ten, which means that if your source file includes the instruction **LD A,10**, the decimal value ten will be loaded into the A register. You can set the radix anywhere from two to sixteen. If the radix were set to sixteen, the instruction **LD A,10** would load the hexadecimal (base sixteen) value of 10H — 16 in decimal — into the A register. (Just as a suggestion, it is a good idea to leave the radix at ten and explicitly specify numbers in other bases.)

Macros

Z80Asm's macro definitions are simple and match the style of most other assemblers. The basic format to start a macro is **NAME MACRO #parameter1, #parameter2, #parameter3…**

The macro name follows the same rules as a program label. Parameters are not required but there may be as many parameters as will fit on a line. Each parameter should be preceeded by a number sign or #.

Macro definitions end with "ENDM" on a line by itself. If macros have been nested (placed one inside another), each macro must end with an "ENDM." If you want to exit a macro definition prematurely (as in some kind of an error condition), use "EXITM" on a line by itself.

Conditional statements can be used inside macros to make them a little more flexible.

Z80Asm also supports the three standard built-in macros. The first is REPT number, which will repeat the macro definition a specified number of times. The second is IRP symbol,<1,2,3,4,5...>, which will repeat the macro definition once for each of the items in the argument list. The third is IRPC symbol,<string>, which will repeat the macro definition once for each character in the string.

Z80Asm —TRS-80 Assembler for Windows

Get the software

At this page one can:

- download the shareware version
- apply for a registered version.

Z80Asm — The registered version

The registered version includes these extra features not available in the shareware version:

- The registered version can generate not only CMD files, but HEX, core image, or COM files as well
- The registered version can create an EQUate listing
- The registered version can directly read from and write to TRS-80 virtual disk files