Tornado by Roelof Koning 1991-2005 v2.1 rev 20-11-2011
========== rev.2016!

A Z80 assembler for the Spectrum, featuring a.o. an intelligent editor and the absence of line numbers.
The short-cut keys, like 'insert' etc, follow the habits of TasWord3. Alas we lost TW3 and these habits years ago.... See the list somewhere below.
Tornado is designed with a facility to 'INCLUDE' sources direct from disk. There are versions for +D, Opus, and HC2000. There also is version v21bis for the IF1bis, where this INCLUDE feature is **shut down** to allow a higher RAMTOP and to make room for IF1 channels.

The assembler/editor can be left for returning to the BASIC environment at any moment by chosing 'B' from the menu. **The menu is found under the 'Extended Mode' key(s) (CS+SS).**
Re-entering in the menu from BASIC is done by typing RUN.
This facility makes it possible to create a simple test environment in a few BASIC lines (20-900 are free). I often call the in-screen disassembler TOPS from line 20... And prepare a RANDOMISE USR statement at line 100 to make test runs of assembled code. (Save your text file before running!)
The source text is built up from address 32768 on (in tokenised format), and the symbol table grows down from 65520, so there usually is plenty of room left for storing assembled code for a test run. I mostly choose ORG 40000 or 50000 for that. (Plus DUMP 40000, of course)
This "all in one" approach is the reason why the text is stored in tokenised format and the menu choices had to be kept 'minimal'.

The **menu page** should turn up after RUN, and can at all times be found under the 'Extended Mode' key combination (CS+SS). This menu provides a number of basic functions for editing, and here also the important 'Assemble' function is found. The Editor is entered by pressing 'N' when starting a new text, or by pressing 'O' for continuing in an existing 'Old' one. The existing text is preserved above RAMTOP and not affected by a RUN.

A **new line** is generated when 'ENTER' is pressed at the end of a line. If not at the end of the current line, then 'ENTER' will just cause a jump to the start of the next line. Empty lines are allowed like in normal text.
As soon the cursor leaves a line then the program will **tokenize** the text of the typed in line and will arrange it into fixed columns. Z80-mnemonics return on screen in capitals. The tokenizing is not case-sensitive, while mnemonics are expected to be found between spaces, commas or '()'.
All spaces are removed from the text during tokenizing, but comments (remarks) following a semicolon ';' are left alone.
The few characters that (on Spectrum) are found printed in red BELOW the keys are not available because of the way the keyboard is read. These rarely used characters cannot be used in comments, but can still be in your code by using a DEFB n(umber) statement.

- The **semicolon** ';' which serves as a remark sign will stick in column 0 as well as in column 32, allowing full comment lines.
- Complete **empty lines** are allowed, but useless empty lines at the end of a file will not stay when the cursor leaves.
- **Strings** should be inside "", not ''.
- **Calculations** can be done with numbers, symbols and the pseudo opcode '$'. Expressions with +, -, *, and / are valid. Note that there is **no priority** in longer formulas, and that no error is generated in case of overflow!
- Numbers in **hex** should be preceded by '#' and **binairy** numbers by '%'.
- **AND** and **OR** functions are represented by '&' and '!'.
- It is possible to have **more than one instruction on a line**, separated by a colon ':'. Spaces following such colon are not removed from the line.
- **Label names** must start with a letter, and can be up to 14 characters long. Names are not case sensitive, although capitals are taken in account when sorting for a View. Label names should NOT end with a colon ':'.
- Complete **empty lines** are allowed, but at the end of the text they will not stay when the cursor leaves.

Editor functions that are available direct under the keys:
(some may recognise the functions of TW3...)

CAPs+1 (**'Edit'**) :start searching in the text. By default this is for the errormarker '(c)' (CHR$ 127), but when a string of text is entered in the 'FIND' menu-function, it takes that. As soon as the end of the text is reached then the findstring returns to the default (c) character. Such (c) character is **default** in the first line, for making a search session begin there.

CAPs+2 :capslock
CAPs+3 :page down
CAPs+4 :page up
CAPs+5,6,7,8 :cursor
CAPs+9 :insert line with **block marker** (CHR$ 126)
CAPs+0 :backspace
Symbshift+ s (NOT) :insert **(c)** marker
Symbshift+ q (<=) :delete one character
Symbshift+ e (=>) :insert one character
Symbshift+ y (AND) :insert one line (**below** the current!)
Symbshift+ u (OR) :delete the current line
Symbshift+ Capshift :**to menu** (normally 'extended mode')

Block-functions (found on the menu screen) like Move and Copy handle the part of the text that is found between two **block markers** inserted with GRAPHICS (CS+9).
The marked blocks are moved to the position where the cursor was on the moment when the 'EXTENDED MODE' keys (CS+SS) were pressed for entering the menu screen.
When blockmarkers are found by the PRINT TEXT function then only the part of the text between the markers will be printed.
Always two block markers are expected. These are removed from the text after the function is executed.

The LOAD, SAVE, MERGE etc. functions on the menu screen are handled in BASIC lines. This way the load/save procedures can be adapted to many systems. Entering an empty file name ("") will invoke a CAT command from BASIC. Check the listing for making your adjustments.

The View and Print Symbols functions return a sorted list of symbols. Every time one of these functions is chosen, the function toggles between an aphabetical sort on the symbol names and a numerical sort on the addresses.
The sort function is case-sensitive, although the assembler is not! This feature makes it possible to move labels that you want to monitor, in front of the listing: just insert a capital in the name.
Known bug: Pressing 'V'or 'P' while the list of symbols is still empty, causes the program to crash!

In most systems stream #3 can be re-directed away from the printer to a file on disk. You then could 'print' the symbol table into a file.
The following is an example using IF1 syntax:
In Basic: **CLOSE #3** [enter], **OPEN #3;"m";1;"filenaam": RUN** [enter]
(Note that this CLOSE #3 may not work the first time when no printer is connected, just 'BREAK' and try again!)
Then choose 'P' or/and(!) 'T' in the menu, and press 'B' when ready.
Then in Basic: **CLOSE #3.**

Errors during assembly are signalled by inserting extra lines. Inserted error messages are always preceeded by a (c) character. These lines can easily be found with help of the 'FIND' function under the EDIT key (CS+1). This function default searches for the '(c)' sign (and also for the end-of-file marker). Intentionally inserting (c) characters (Symbsh+'s') would provide you with a smooth way to go to relevant parts in long texts.

The **end-of-line character** is CHR$ 13 and the end-of-file is CHR$ 255. CHR$ 126 is used as blockmarker, and CHR$ 127 serves as errormarker (c).
The **text file** grows up from memory address 32768, while the symbol table grows down from 65520. Possible collisons between the both are detected and signalled before they actualy take place.

PSEUDO OPCODES:

```
;-------------------------------------------------------------------------
```

Apart from the usual **ORG, EQU, DEFB, DEFW, DEFS.**

```
DUMP       : Address where the assembled code is placed. Default value =0,
             then (and also when omitted!) no code is generated.
             The program will only allow dumps to free memory addresses
$          : Represents the value of the current address (Program Counter)
END        : The assembling will ignore any text following this opcode
INCLUDE    : Syntax is: INCLUDE drivenumber "filename".
             Allows to assemble files from disk. This allows for sources
             of >20Kb to assembled. Or multiple sources could generate a
             code block >10 Kb! (Not present in v.21b!)
```

Because of this 'include' function there are **different versions** of Tornado for a number of interfaces. **Proceed with care!** The file to be included is loaded in a part of the screen and assembled from there. An invalid drive number is signalled in the main text with error message **"number?"**. Missing quotes are reported as **"nonsense"**. A nonexisting filename, or a non-CODE file, or a CODE file that has a start address that differs from 32768, is signalled by a jump to the Spectrum's **"Invalid filename" error.**
Recursive including is not allowed (a text to be included cannot hold an include instruction), this will result in jump to the Spectrum's **"Statement lost" error.**
The code that executes the "include" directive is the **only** disk-system specific part of the program. When this directive is not used, the code plus the eventually modified basic will run on all Spectrums and emulators.

Final:

There are a number of text **convertors** available from several authors: gens2tor, tor2wm, tor2txt, txt2tor, mem2defb.

Technical hint: An existing disassembly text in plain ASCII can be used as source file after all 'garbage' (including all CHR$10's!) is removed or changed into CHR$32's (space). Make sure(!) that such text ends with CHR$13 + CHR$255.
You can make Tornado believe that all lines must be **re-tokenized** by poking 28637,0. Walking through the text by pressing 'Enter' then does the job.
Do not forget to restore the normal situation by poking 28637, 208!

Roelof Koning, 2005.
Rewritten for v2.0 for HC2000
Revamped 2016