I have been using Zen for over a year (as well as Zeap), and have patched Zen to run on Nas-Sys. It is a very fast assembler with one drawback - a lousy error handling function ! It aborts assembly on the first error it finds. In "Writing Interactive Interpreters and Compiliers", P.J. Brown points out that a compiler spends most of its time in the error handling mode, and should be designed to do this well - Zen definitely doesn't. I have been tidying it up, and have written a number of Pseudo ops; TITLE to title a printed page, LIST and UNLIST to allow selective printout of sections during assembly. RCAL and SCAL to make life under Nas-Sys easier. I have also taken the DB, DW, DM, pseudo ops to Zilog/Mostek standards (DEFB etc). My next miracle will be to tidy up the error handling, so that it will do the first pass, and point all first pass errors. Then (perhaps) a macro facility ?

STOP !....(please?)

HALT !

By Richard Beal

======

If you want to halt a program this can be done by including the code 76 in your program. When this HALT instruction is executed, the program counter stops being incremented and an endless stream of NOPs are executed. The Z80 CPU detects that it has halted and a LED on the Nascom lights to show that it is in a HALT state.

There are two ways to leave a HALT state. The first and most commonly used is to press Reset, which restarts the computer and reinitialises Nas-Sys.

The second method is for there to be an interrupt. In fact, if you were writing a program which was to do nothing at all except when an interrupt occurred, you could just code a HALT instruction. Note that if you intended the Z80 to re-enter the HALT state after the completion of the interrupt, then the code would be as follows:

76 18 FD

as the address of the HALT (pushed on the stack) at the start of the interrupt, is POPed off and incremented when a RETI instruction is encountered.

This explains why you can Single-step through HALT instructions. The HALT is executed once, but the NMI (Non Maskable Interrupt) generated by Single-stepping jumps out of the HALT state at the end of the instruction.

If you try to execute a program which has a HALT instruction as the first byte, you will find that it will not HALT. The reason is that the Execute command in fact Single-steps the first instruction and then executes normally. So if the program was:

76 .. .. .. it would not HALT

But if it was:

00 76 .. .. it would HALT.
