Working with ZXAS

What is ZXAS?

ZXAS is an assewmbler for the Sinclair ZX81. It allows you to write and assemble z80 assembly code directly on the ZX81, without the need for a seperate computer (Windows/Linux PC or Mac). This gives you the full rfetro feel, but be prepared, it is harder to write big programs, and a lot more has to be worked out (designed) on paper before/during writing the code.

Getting started

Download ZXAS

<https://www.timexsinclair.com/product/zxas/>

or <https://www.zx81stuff.org.uk/zx81/tape/ZXAS>

There are a few ways to work with ZXAS, for small programs the first is the easist to get something running. The following assumes you are running on a real ZX81, not an emulator. If you are using an emulator then it's all much easier, you save snapshots of RAM etc to preserve the machine code, otherwise on a real ZX81 it is lost when the power is turned off. In fact with some emulators you can't even load in a tape without it doing a reset!

Method 1 - small code straight away, not worried about saving

1) Load ZXAS from tape, or TZXduino - this loads a basic program which has a big REM statement at the start - that is the ZXAS assembler in a single basic line. The memory for a "REM in basic encoded machine code program" starts executing at 16514.

2) run the program by pressing "R" to key in "RUN" - this "installs" the assembler into memory, the first thing the code does is copy the assembler to memory location 28565.

3) delete lines 10, 20 and 30. This then allows upto 8999 lines of assembly code!

4) your program can now be entered as REM statements in the basic editor using the ZXAS assembler syntax. This is pretty much as per standard z80 assembler, but there are a few differences. Most notably the comma is replaced by a full stop, and this makes it eaiser to type in code on the ZX81 keyboard (comma key would have to be SHIFTED to access).

example program - put a character on screen using ROM routines



Notes:

Program must start with "REM (" and end with a line "REM )".

Numbers can be entered in hexadecimal and preceed with "$", or decimal.

Comments can be added after instruction or on own line with "\*"

Always end program with RET to return to basic.

5) Now assemble the code by running command "GOTO 9000"

the L prompt requires uyou to enter a memory address in decimal - 25000 a decent place for this code. If you wanted to save the prgram after you would not be able to use this without first copying it back to a REM statement in the basic area. This is easy todo with an additional basic program. If there are no errors then you see the assembled code. For some reason the code is repeated as displayed. This might be due to multi pass to resolve lables:



Notes:

The first column is the basic line, the second is the memory address that is is assembled to, third the machine code instruction and the last column is your assembly code.

6) You could just run your code by typing RAND USR 25000 but this would cause it to run off the end into the line 9000 when the assembler is called from. Its best to type in another line of basic code - "8999 STOP", then RAND USR 25000 without a line number:



The result is a little graphic symbol which represnets character 6 being printed in row 5 of the 10th column:



Method 2 - larger programs and have to assemble each time

1) Load the ZXAS program as before but don't

2) delete lines 10, 20, and 30

3) save the program as it now stands to a new file - say "MYCODE"

4) Load and run the ZXAS program to "install" it at 28565

5) now type and run the NEW command - this will clear the whole basic program

6) load MYCODE from tape again - without resetting the ZX81

What we have now is the assembler in memory at 28565, we have the basic program including the code that runs the assembler (GOTO 9000), and we can freely type in our program and save it again as many times as we like. We can assemble it and run it as well at any time. The catch is, to run our code we alwasy have to load the assembler then NEW, then laod our code then assemble. OK for dev but not if releasing the program to the world.

Method 3 - for distributing [larger] programs: assemble once, copy back to basic and save

As mentioned before to save the assemble code you have to get it back into a REM statement at the start of the basic program. There are two ways todo this.

1) If you know the size in bytes of your assembled code then it is possible to create an initial REM statement on say line 1 with enough characters to hold the assembled code. Put this before any assembly code is typed in. Now you will be able to assemble to 16514 when prompted by the assembler (rather than 25000 in the example).

2) write a small bit of mnachine code to copy the assembly code into the line 1 REM statement after it is assembled at 25000 (or wherever). You can then save the program to tape.

For more information see book Assembly Language Assembled for the Sinclair ZX81 by Anthoy Woods. I've tried to obtain a paper copy but unfortunately coulnd't find any on ebay. It is preserved at:

<https://youbookinc.com/pdf/assembly-language-assembled-for-the-sinclair-zx81/>