YOUR COMPUTER

ZX-81 Assembler / “Super'Sembler”

by *Anthony Nwokoye*

Your Computer / November 1984 (Pages 141, 143 and 144)

This program was written for ZX-81 owners who would like to move on to machine code but who do not want to buy all the necessary assemblers, monitors, etc. It only occupies 3.75K of memory, above RAMtop, and has features you would expect to find the top commercial products. These features are:

- ***The ability to assemble any Z-80 instruction – pages 181 to 187 of ZX-81 Basic Manual.***
- ***It assembles to any part of the memory.***
- ***It handles all labels, including those which require a 16-bit address.***
- ***It accepts decimal, hex, or binary numbers.***
- ***The ability to have messages imbedded in your code by putting the message between quotes.***
- ***Comments and reminders may be placed after an asterisk. You can also have multiple instructions, with each separated by a semi-colon – the whole code could be assembled from one line.***

With these facilities, it would make it easier to write fast and smooth arcade-type games, and improve your machine-code techniques.

The machine code itself is stored in a Rem line 3,871 bytes long. This is obtained by first entering:

1 REM 221 characters

Then edit line and change it to 2 and edit 2 and change it to 3 and so on, until you have 1 to 17 lines.

Then add

18 REM 7 characters

Then you **POKE 16512, 15**

POKE 16511, 28

then you enter the hex loader. Once you're entered the code and saved it to tape you can try out the assembler.

Load the program. When it has been loaded you should list it. You will see a long Rem statement followed by a mixture of weird characters. This is the machine-code assembler and the computer is only listing the first 400 bytes or so of the code. This code needs to be about RAMtop, so to do this you type:

PRINT USR 16514 …(Newline)

and then you will see the computer New itself as it sends the code above RAMtop and clear itself.

Now that the computer is clear and the machine code is above RAMtop, you are ready to enter your mnemonics. I am going to have difficulty explaining this, so please stay awake!

Before entering your mnemonics, you need to enter a Rem line that is suitable to hold the code. The code does not have to start at 16514, it can start at, say, 22000. Before any mnemonics are entered you need a Rem line with an opening bracket, so the assembler knows where the code begins such as,

1 Rem… necessary bytes

10 Rem (open brackets

all mnemonics are entered in Rem lines like so:

1 Rem …

10 Rem (

20 Rem LD A, 0

to have multiple statements use a semicolon:

1 Rem…

10 Rem(

20 Rem LD A, 0; RST 16; BIT 7, (HL); etc.

The use of labels makes any assembler, without doubt, much easier to use. They are used in instructions like JR or DJNZ, and you can even have 16-bit labels where the computer fills in the necessary two bytes. To enter a label you have to type a colo followed by an L and then the label number, and then again a colon. You have to do this before the required statement, like so:

1 Rem…

10 Rem (

20 Rem L1: INC HL; JR L1

The label number can be anything from 0 to 255 and must not exceed these values as the labelling system won't work correctly.

Now for the 16-bit label. In an operation like so:

1 Rem…

10 Rem (

20 Rem :L0: SET 7,A; XOR A etc…

30 Rem LD BC, L0 – here, BC is loaded with the 16-bit address of where label 0 would be when the mnemonics are assembled, or

20 Rem : L0: SET 7,A; XOR A , etc…

30 Rem Call L0 – the address is automatically done when assembled. In all the examples I've used decimal numbers, but I could easily use hex or even binary numbers. To enter a hex number, you must put an asterisk

before it like so:

LD A, *2A, or LD DE, *B2CA

For a binary number, you put a plus sign before it like so:

LD, HL, +01101010000101, or

LD C, +10100011

One of the special features of this assembler is that you can have messages imbedded in your code. These can also be labelled and are put between quotes. Here is an example:

1 Rem…

10 Rem (

20 Rem "THIS IS A MESSAGE"

and label the same like so:

1 Rem …

10 Rem (

20 Rem :LS: ”THIS IS A MESSAGE”

30 Rem LD HL, LS; LD BC, etc.

To have reminders and comments in your mnemonics, so you know which part does what, you use an asterisk. This is put before the reminder.

Like so:

1 Rem ...

10 Rem (

20 Rem *THIS IS A COMMENT

30 Rem LD HL, 2A0K; etc

40 Rem * THIS IS etc and so on.

Now when you’ve finished your mnemonics, you enter a Rem) close brackets. This has to be at a line, at the end of your mnemonics. This is so the assembler knows when the end of assembling has been reached. Then you type in this short program to run the assembler:

9990 LET ADD = 16514

9991 POKE 32767, INT(ADD/256)

9992 POKE 32766, ADD – 256 * INT(ADD/256)

9993 LET A = USR 27819

The value of Add doesn’t have to be 16514, but can be any address where memory is reserved for the code, except 27819 or upwards as this holds the assembler and the label stack. Both address 32766 and 32767 will hold the address of the place the assembler should dump to.

Now for the moment of truth. Type Run. You should see the screen flicker and at the top of the screen:

LOOP ACCOMPLISHED

Should have been written. This means that if you now List the program, you will see at line number 1 the compiled machine code, and all is well.

However, if the computer prints:

I **CAN'T ASSEMBLE ONE OF THE CODES**

Then it means that you've made an error with one of the mnemonics. To help you find the error the computer displays at the bottom of the screen a 9 followed by a / sign. After this is the number logo where the mistake occurred so:

9/5

Would mean a mistake has happened at loop 5.

If the computer prints:

YOU HAVE MISSED A LABEL USING JUMP

It simply means that you have in the mnemonics requsted a label using JR, DJNZ, etc. when this label doesn't exist.

YOU HAVE MISSED A LABEL USING CALL would mean that you've requested a non-existent label, maybe using CALL L92, LD HL, L12, etc.

There must only be one space between the command and the next number or register in the mnemonics. For example LED (65535), A is acceptable, while LD (65535), A is not. You use a full stop or a space to separate each section. For example, LD A, B is good , as is LD A B. All RST should have their numbers in decimal and have the numbers close to the letters like: RST48. All label numbers should be in decimal, and there should be no space at the end of the line.

From DesSony (December 2020).