

# ZX-CESIL

T. Gilberts



## C O N T E N T S

INTRODUCTION	2
LOADING ZX - CESIL	2
STARTING CESIL	3
DATA, IN & OUT	4
ARITHMETIC	6
LOAD & STORE	7
JUMPS AND LABELS	8
LINE	9
DEVELOPING PROGRAMS	9
CASSETTE OPERATION	12
PRINTER OPERATION	12
USE OF *BASIC	12
SUMMARY OF ZX - CESIL	13
EDITING (48K ONLY)	14
NOTES FOR ZX81 USERS	15

## INTRODUCTION

CESIL (Computer Education in Schools Instruction Language) was developed to teach the rudiments of programming to computer novices. Usually at 'O' level the programs are written onto coding sheets and sent away to be entered into a computer. This method does not really enable a familiarity to be gained with the language and so we developed ZX - CESIL, to allow the same programs to be run on a Spectrum.

As this program is designed to run on either a 16k or a 48k Spectrum it does not include any editing features due to the memory restraints. A 'BASIC' command has been included to allow simple tasks to be carried out from BASIC.

## LOADING ZX - CESIL

Two copies of the program are recorded on the high quality cassette included with this manual. To ensure they will LOAD they have both been VERIFIED. Position the tape just after the leader and type the following into the Spectrum;

LOAD "cesil" and then press 'ENTER'

start the tape recorder on PLAY, (the Anti-record pips have been removed for safety.) If after the two tone bars have come onto the screen the program has not been recognised (no blue and yellow lines) then rewind and try again with the volume turned up slightly more.

When it has LOADED it would be wise to make a copy for day to day use to preserve the original. (see the section on cassette operation.)

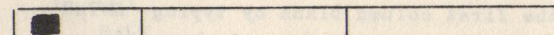
If the program has LOADED you should briefly see the following

SPECTRUM CESIL - PLEASE WAIT

and then the screen appears thus;

ZX-Cesil

va. 1



You are now ready to start using ZX - CESIL.



## STARTING ZX - CESIL

The screen display is a simulation of the three columns of a CESIL coding sheet, that is;

LABEL      INSTRUCTION      OPERAND

this will make typing in a program very easy. For the moment we can ignore the first column. The second, 'INSTRUCTION' column is the one which contains the actual commands that we give the computer. The third column gives the information for it to use. Lets give it a simple program to demonstrate.

PRINT "HELLO THIS IS ZX-CESIL"

HALT

The purpose of this program is to print out on the screen the message 'HELLO THIS IS ZX-CESIL' (the PRINT statement causes this to happen) and then stop executing instructions (the HALT statement).

To enter this into the computer carry out the following; The first column is empty so move the cursor to the second column by pressing the 'ENTER' key, (the cursor is the flashing C in the bottom of the screen). If you need to type something in the first column the cursor can be moved back by pressing 'ENTER' again, before you type anything else. Unlike Spectrum BASIC you must type the word PRINT in full. You may use all the normal cursor keys to correct spelling etc, and when you are satisfied press 'ENTER'. The cursor will then move to the third column or in this case print the prompt,

PRINT TEXT:

Now type in the message either with or without the inverted commas. When you have finished press the 'ENTER' key again. (from here on it can be assumed that 'ENTER' will be required after every entry.) The entire line will be printed on the upper part of the screen and the cursor moved to the first column. Now enter the HALT statement in a similar way to above, but leave the third as well as the first column blank by typing 'ENTER'.

The computer has now stored this program in memory, it will do nothing with it until instructed by you. You can do this by telling it to 'RUN' the program, so type;

\*RUN in the first column and press 'ENTER'

The computer will now carry out the instructions and produce the following display;

HELLO THIS IS ZX-CESIL

\*\*\* HALT statement, NO ERROR \*\*\*

This shows it has carried out the program successfully, try it again if you like.

To introduce another command type;

\*LIST in the first column and press 'ENTER'

The computer will now print out a listing of your program.

## DATA, IN & OUT

Get rid of the above program be typing;

\*NEW in the first column and press 'ENTER'

This will cause the computer to clean out memory of all information by starting up ZX-CESIL just as if you had reloaded it.

All of these words in the first column are commands and cannot be used as part of a program, only to start the computer doing something while they are preceded by an asterisk '\*', if you leave out the asterisk the computer will accept them as part of a program, see later on.

The next part of CESIL to introduce is DATA. DATA is a string of numbers, positive or negative but not decimal, (you cant have a decimal point!). To illustrate the use of DATA enter the short program which follows.

LABEL	INSTRUCTION	OPERAND	PRINT TEXT:
	IN		
	PRINT		"THE NUMBER IS "
	OUT		
	HALT		

Now type RUN as above. The program stops with;

\*\*\* Out of DATA ERROR \*\*\*

This is an error report, and tells you that the computer had no information. It required this when it met the IN instruction.



The IN instruction is causing the computer to input a number from the string of DATA, but we haven't given it any! so type;

\*DATA as a command.

and the computer will ask;

How much data

we only want to give it one piece so type '1' and press 'ENTER', it will then prompt;

value 1 =

Type in any whole number you like here and press 'ENTER'.

Now RUN the program again, this time it produces;

THE NUMBER IS 'whatever number you entered'

\*\*\* HALT statement, NO ERROR \*\*\*

When this program is RUN the following happens, the IN instruction causes the computer to fetch the first piece of DATA which is your number, and store it in the accumulator. The accumulator is a space in the computer which can hold a single number. The PRINT instruction causes it to print out the message THE NUMBER IS . The OUT instruction causes it to print out the value of the accumulator, and the HALT you already know.

Try giving the computer different DATA and then RUN the program again.

To show that the accumulator will hold only one number try the following;

IN  
IN  
IN  
OUT  
HALT

Enter the following data, 10,-34,8. As an aside LIST the program and you will see that the computer gives a list of the DATA as well as the program.

On typing RUN the computer will produce;

8

\*\*\* HALT statement, NO ERROR \*\*\*

The reason for this can be shown by the following diagram;

INSTRUCTION	ACCUMULATOR CONTENTS
IN	10
IN	-34
IN	8
OUT	9
HALT	8

This shows that each IN instruction wipes out the previous contents of the accumulator.

### ARITHMETIC

CESIL can carry out all the normal arithmetic operations on the numbers it holds in the accumulator. Type in the following program;

IN  
ADD +1  
OUT  
HALT

Note that the '1' goes in the third column and must have a '+' sign in front. Enter a DATA number and RUN the program, it will print out your number plus one. The ADD instruction did this by adding the number following it ('+1' in this case) to the contents of the accumulator (your number).

The SUBTRACT instruction does a similar thing except it takes away the number. As a test rewrite the above program but use SUBTRACT instead of add, and a different operand, (the number which follows).

MULTIPLY will cause the number in the accumulator to be multiplied by the number following the instruction. Note that the result is still in the accumulator overwriting the original number with all of the arithmetic instructions.

The DIVIDE instruction is slightly different. It will only produce whole number results, so that 5/2 produces 2 rather than 2.5, try writing a program to do this calculation.



## LOAD & STORE

It was mentioned earlier that the accumulator could store only one number, but many problems need to consider quite a few, so how do we do it?

The answer lies in a storage location which we can cause the computer to set up, and then refer to it by an identifier or name. An identifier can be up to six characters long and consist of any sequence of letters and numbers, as long as it starts with a letter. Thus all of the following are legal identifiers;

```
LOOP
L9
ANCHOR
NOT ME
A142
```

We use the STORE instruction followed by the identifier to set up the location containing the current value of the accumulator. This identifier can then follow any of the arithmetic instructions in place of the number (its called a constant), and the value it contains will be used by the computer when it executes the instruction. Try the following program which carries out the 5/2 calculation posed above;

```
IN
STORE NUMBER
IN
DIVIDE NUMBER
PRINT "RESULT = "
OUT
HALT
```

DATA 2,5

There is a far simpler way of carrying out this operation, but this program demonstrates the use of storage locations.

There is another instruction, not considered yet, which can be used with a storage location. This is the LOAD instruction which is used to copy the contents of a storage location to the accumulator, or it can be followed by a number and thus place that into the accumulator.

The following program shows the two uses of the LOAD instruction. Note it is not intended to carry out any other task.

```
LOAD +100      put an initial value
STORE TEMP     into location TEMP.
LOAD +0        clear out the accumulator.
OUT            prove its been done.
LOAD TEMP      put the value back into
OUT            the accumulator and show it.
HALT          stop the program.
```

When this program is RUN it will produce the following output:

```
0 100
```

\*\*\* HALT statement, NO ERROR \*\*\*

Showing that TEMP held the value in the accumulator when the STORE instruction was executed and that it was put back in when the LOAD TEMP was executed.

## JUMPS AND LABELS

It is fairly easy to write programs which run through an instruction at a time and then stop. But, the real power of the computer is not revealed until it repeats parts and decides what to do on a particular result. This is referred to as branching. To do a branch we need to use the first column, to give the line a label which will allow us to refer to it. A label can be up to six characters in length, must begin with a letter but may continue with any combination of letters and numbers.

The following program has a label on the PRINT instruction.

```
LOOP PRINT "HELLO"
      JUMP LOOP
```

Enter this and RUN it. It will keep on printing HELLO's one after another (to stop it press the SPACE key and it will stop with;

\*\*\* No HALT statement ERROR \*\*\*

The program is what is known as an infinite loop, as when the computer comes to the JUMP instruction, it moves to the line that has the same label as the one following the JUMP instruction, which in this case is the beginning of the program so it keeps going round and round in a loop.



What we really need is a controlled loop, one that knows when to end. There are two instructions in CESIL for doing just that, they are;

JINEG label - Jump to the line with the given label if the contents of the accumulator is negative.

JIZERO label - Jump to the line with the given label if the accumulator is zero.

The following program uses the JINEG instruction to end the loop when a negative number is put into the accumulator, If it is not a negative number the value is printed.

The program also introduces the LINE instruction, this sets the next position for CESIL to PRINT on the screen at the beginning of the next line. You may have noticed that normally any output is placed straight after any previous output, try the program without the LINE instruction to clarify this.

```
*LIST
PROGRAM LISTING;
LOOP      IN
          LINE
          JINEG    END
          PRINT    "THE NUMBER IS"
          OUT
          JUMP     LOOP
END        PRINT    "END OF DATA."
          HALT

% DATA;
10, 76, 0, 4, 3462, 1, -1
```

The JIZERO instruction can be used in a similar way to the JINEG instruction above. In fact the next section provides a good example of its use.

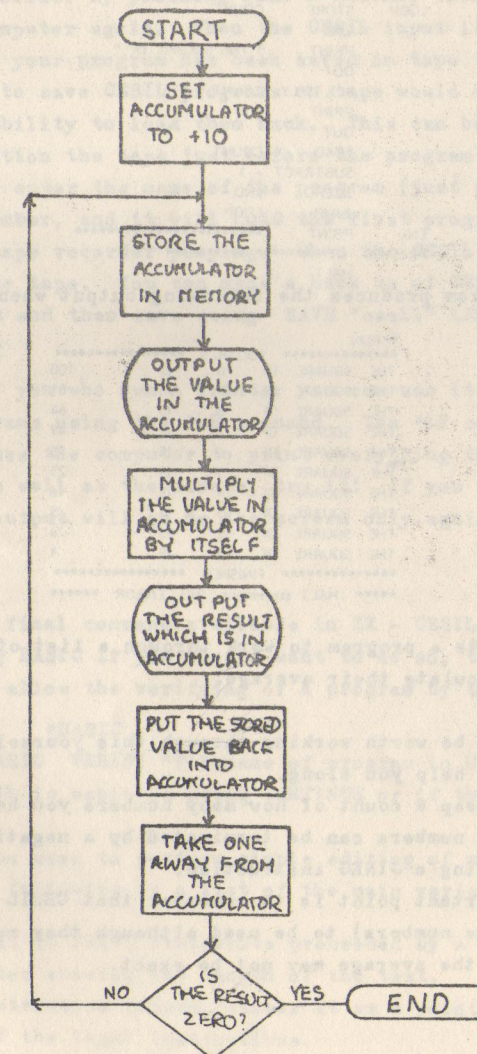
#### DEVELOPING PROGRAMS

All of the programs given so far have been extremely simple used only to illustrate a point in the text. CESIL is a very low level language, in fact it is meant to demonstrate the use of assembly language. There is a great deal of fun to be gained from using such languages to try and solve simple problems, which could be done in a high level (more advanced) language such as BASIC much easier. This thus requires more advanced programs and demands a logical approach to the problem. As an example of problem solving using CESIL a worked example will be given.

PROBLEM : Using only two constants and a single storage location write a program to output a table of the numbers and their squares from 1 to 10.

The first step in solving this is to think about what we are being asked to do; there must be a controlled loop, a single memory (storage location), two constants (numbers) and squaring can be achieved by multiplying the number by itself.

The next step is to write down the solution as a Flowchart.





Apart from a couple of clarifying PRINT instructions the program follows the flowchart exactly. You might like to try coding it yourself first.

\* LIST

PROGRAM LISTING:

```

SQUARES  PRINT *****
          *****
          LOAD  +10
          STORE COUNT
          LINE
          PRINT " THE SQUARE OF "
          OUT
          MULTIPLY COUNT
          PRINT " IS "
          OUT
          LOAD  COUNT
          SUBTRACT + 1
          JIZERO END
          JUMP  LOOP
          END  PRINT *****
          FINISHED *****
          HALT

```

The program produces the following output when RUN.

```

* RUN
***** SQUARES *****
THE SQUARE OF 10 IS 100
THE SQUARE OF 9 IS 81
THE SQUARE OF 8 IS 64
THE SQUARE OF 7 IS 49
THE SQUARE OF 6 IS 36
THE SQUARE OF 5 IS 25
THE SQUARE OF 4 IS 16
THE SQUARE OF 3 IS 9
THE SQUARE OF 2 IS 4
THE SQUARE OF 1 IS 1
***** FINISHED *****
**** HALT statement, NO ERROR ****

```

PROBLEM : Write a program to work through a list of numbers and calculate their average.

It would be worth working through this yourself, so here are some hints to help you along. Remember to keep a count of how many numbers you have totaled. The string of numbers can be terminated by a negative number and thus found using a JINEG instruction. The most important point is to remember that CESIL will only allow integers(whole numbers) to be used although they may be negative or positive, the average may not be exact.

## CASSETTE OPERATION

ZX - CESIL includes the ability to save programs onto cassette, using the SAVE command. Type in a reasonable length program and find a clean section of tape. Type \*SAVE and you will be prompted

File Name ?

Enter any name up to 10 characters in length and press 'ENTER'. Start the tape recorder by pressing PLAY & RECORD, then press 'ENTER' on the computer again. When the CESIL input line appears stop the tape and your program has been saved on tape.

The ability to save CESIL programs on tape would not be any use without the ability to load them back. This can be achieved as follows; position the tape just before the program you want, type in \*LOAD and enter the name of the program (just press 'ENTER' if you don't remember, and it will LOAD the first program it comes to). Start the tape recorder playing. When the CESIL input line reappears stop the tape. You can make a back up of CESIL thus; \*BASIC then CLEAR and then save using SAVE "cesil" LINE 9000.

## PRINTER OPERATION

For those of you who own a Printer you can use it to print out results and programs using the \*LP command. The \*LP command when first issued causes the computer to print everything that it does, on the printer as well as the screen, try it! If you issue it a second time the output will go to the screen only, again.

## USE OF \*BASIC

This is the final command available in ZX - CESIL, it provides a way of entering BASIC if you should want to do so, the command could be used to allow the verifying of a program by typing

\*BASIC

and then in BASIC VERIFY "filename of program in UPPER CASE" return to CESIL is achieved using CONTINUE or if that fails GOTO 6000.

or \*BASIC could be used to perform simple editing of mistakes in a program, the following is a list of the main variables used;

- f\$ - the text of PRINT statements preceded by a single character showing the length of the text.
- p\$() - the instruction column, stores it as a pointer into z\$
- z\$ - list of the legal instructions.
- x\$ - the label list
- x() - shows where label is.
- c\$ - operand column



## SUMMARY OF ZX - CESIL

The following two tables give a summary of the commands and instructions available in ZX - CESIL.

**** INSTRUCTIONS ****		
Instruction	Operand type	Summary of effect
IN	none	Input next data value to accumulator.
OUT	none	Print out accumulator.
LOAD	iden/cons	Copy value of operand into acc.
STORE	iden	Copy accumulator to given memory.
ADD	iden/cons	Add value of operand to acc.
SUBTRACT	iden/cons	Subtract value of operand from acc.
MULTIPLY	iden/cons	Multiply acc. by value of operand.
DIVIDE	iden/cons	Divide acc. by value of operand.
JUMP	label	Go to the line with the given label.
JINEG	label	Go to line if acc. is negative.
JIZERO	label	Go to line if acc. is zero.
HALT	none	Stop executing CESIL instructions.
PRINT	string	Print out given string.
LINE	none	Start new output line.

iden (identifier) - is the name of a storage location, starting with a letter and followed by up to five letters and numbers. n.b. also for label

cons (constant) - a number, integer only, preceded by either a '-' or a '+' sign. zero is +0.

**** COMMANDS ****	
Command	Effect
NEW	Clear out all memory for CESIL and restart.
DATA	Enter a string of numbers (omit '+' sign here).
LIST	List out current program and data.
LOAD	Load a program from tape.
SAVE	Save a program onto tape.
RUN	Start executing CESIL program from first line.
LP	Echo consol output to printer.
BASIC	Return to BASIC.

All of the commands must be typed in the first input column and preceded by an asterisk '\*'.  
13

## THE \*EDIT COMMAND (48k only)

The 48k version of Cesil (on the reverse side) provides an editor. This can be accessed by typing;

\*EDIT in the first column and 'ENTER'

The editor will then give its version number and the prompt;

>C

with the flashing 'C'. The simplest of the commands is 'R' if you type R and press 'ENTER' you will be returned to normal Cesil operation, R is short for RETURN.

Try typing in the squares program on page 11, and then move into EDIT. The next command for EDIT is 'L', this will produce a listing of the Cesil program, but with a number in front. The other two EDIT commands use these line numbers to refer to lines of program.

The next command is 'I' (INSERT) which allows you to insert an extra Cesil line. For example, to insert a 'LINE' instruction between lines 1 and 2, type I2 and 'ENTER' then you will be given the normal input grid, so type your program line as normal. When the EDIT prompt returns, 'L' the program and the new line has been inserted in the correct position.

This extra line can be deleted by using D(DELETE). In this case the new line has become line 2 so type D2 and then L, you will find the line has been removed.



## NOTES FOR ZX81 USERS

This manual was written to be a teaching guide for Spectrum CESIL. The ZX81 version provides the same functions but there are a number of differences;

- 1/ The ENTER key on the Spectrum is the equivalent of the NEWLINE key.
- 2/ Pressing the SPACE key will return you to BASIC if this happens;

type GOTO 6000 'NEWLINE' to recover.

This also means that the CESIL break key mentioned on the bottom of page 8, has been changed to the NEWLINE key on the ZX81 not the SPACE key as on the Spectrum.

- 3/ All the lower case output in the manual is in upper case on the 81.
- 4/ The \*LP command will cause the screen to flicker violently.
- 5/ The SAVE and LOAD commands do not prompt for a file name but execute immediatly. So start the tape running before pressing the NEWLINE key. The routines save only the CESIL program at twice the speed of BASIC.
- 6/ If your program is longer than 16 lines, CESIL will wait for you to press a letter before it carries on LISTing.
- 7/ The LOADING of ZX81 CESIL can be achieved on the ZX81 as follows;  
Position the tape just on the silent leader. And type;

LOAD "CESIL" then 'NEWLINE'

Now start the tape recorder playing. If it fails to LOAD try adjusting the volume or using the other side of the tape. It is advised that a copy for day to day use be made using RUN 9000.

- 8/ When entering PRINT TEXT: all the normal cursor controls will work. The inverted commas of the PRINT instruction are provided do not rub them out or add any more.
- 9/ On input with the box graphics, the RUBOUT is the only editing key which will work. If you try to RUBOUT beyond the limits the cursor will move to another field.