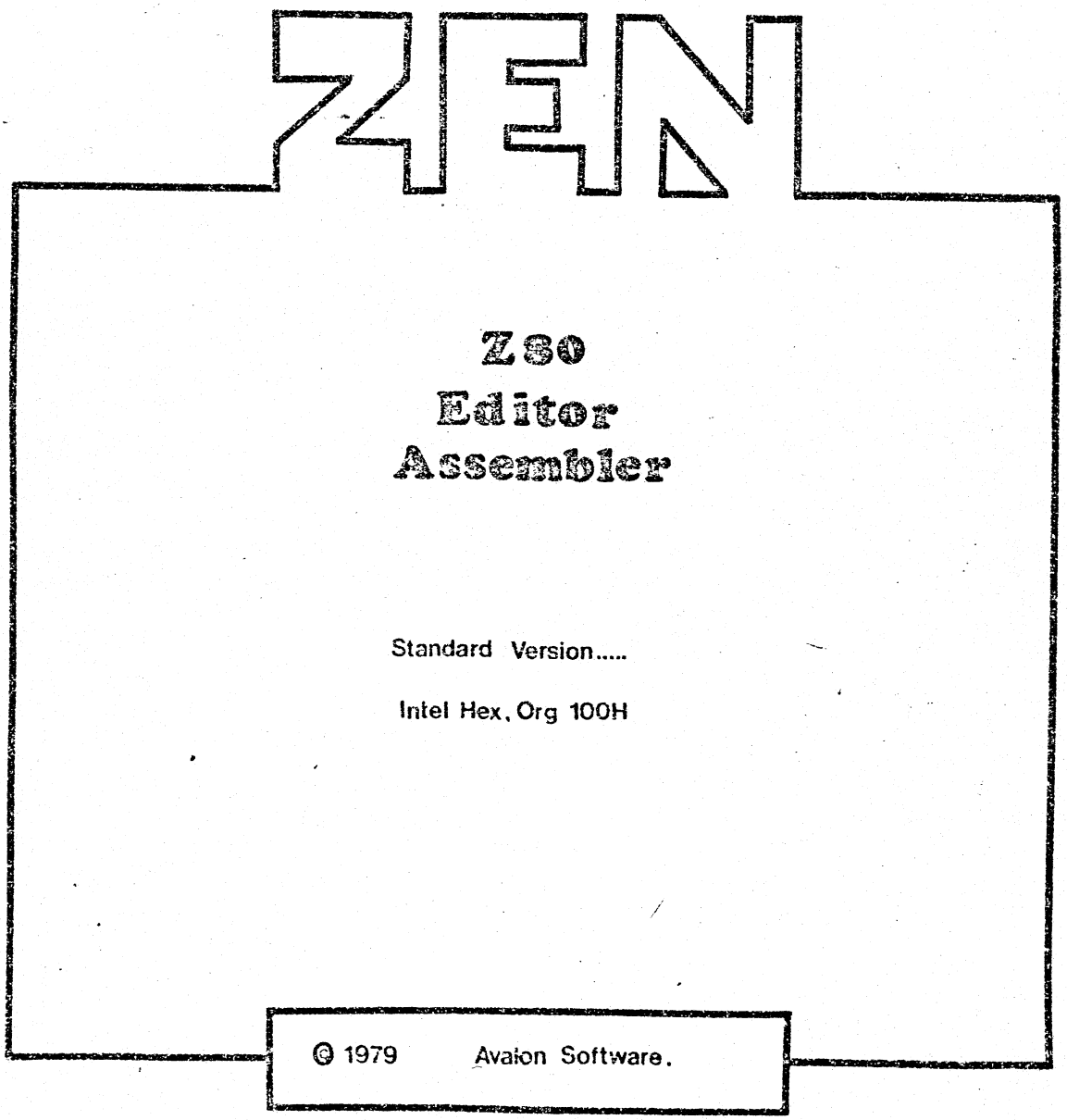
04 = 1B
05 = 1C etc.
06 1D
07 1E

100
8000



Z80
Editor
Assembler

Standard Version.....

Intel Hex, Org 100H

What does an Editor/Assembler do

An ASSEMBLER .... Takes as input a source file and generates either an object file or a listing.

A source file is a series of statements in the Z80 Assembly Language. It is held in memory as pure text.

An object file is a block of runnable Z80 machine code. ZEN can dump the object file to cassette, to memory or to both.

A listing is a record of both source and object code in human-readable form.

An EDITOR ..... Is the means by which the source file is initially created and then altered.

What have I bought

This manual.

A full listing, which was produced by ZEN assembling it's own source file. This source file is available on cassette for an additional charge. It occupies about 20K.

The object cassette produced during assembly. The object file is recorded at 300 baud CUTS in the INTEL HEX format.

Starting up

Unless you are running under ZAPPLE you will need to modify certain sections of code. These modifications are described on pages 12-13.

Entry to ZEN is at 100H (all restarts are free). After initialisation ZEN will display the command loop prompt:

```
z>
```

ZEN is waiting for a command.

COMMANDS

Global:

H .... HOWBIG
C .... CLOSE
R .... READ
W .... WRITE
Q .... QUIT
S .... SORT (Alphabetic parameter)
A .... ASSEMBLE

X .... XCHANGE
V .... VERIFY (TAPE)

Pointer movement:

T .... TOP
B .... BASE
D .... DOWN (Numeric parameter)
U .... UP (Numeric parameter)
P .... PRINT (Numeric parameter)
L .... LOCATE (String parameter)

Edit:

Z .... ZAP (Numeric parameter)
E .... ENTER
N .... NEW

As the Z80 Assembly Language is entirely line orientated the editor in ZEN is line,rather than character,orientated.

ZEN always maintains an internal pointer to the current line. Apart from the global commands all commands either alter your position in the source file or change the contents of the file.

Several commands accept a parameter immediately after the command letter. These are explained in detail later on.

All user input to ZEN is terminated with the Carriage Return (CR) key. You may edit any entry you make with the Backspace key (ASCII 8, usually Control H).
ZEN won't allow input lines to exceed fifty-nine characters.

The null command,that is just typing (CR), will generate a screen clear.

PRE-10H

SCP
RET NZ
JP IMEED

GLOBAL COMMANDS

H .... HOWBIG, displays the Start and End of File (SOF,EOF) in hex.

C .... CLOSE, erases the file in memory,be sure you mean it!

R .... READ, reads a source file from cassette. The tape file is appended to the end of the memory file allowing source modules to be merged. The header message on the file is displayed first for identification. The file is formatted into blocks, if any block contains a cksum error an asterisk is displayed.
If the file overflows available memory the error message MEM is displayed and loading stops.
After loading the new file size is displayed.

W .... WRITE, saves the entire source file on tape in a binary format. Upon entry you will be prompted for a header message. You may enter any message, eg CHESS VERSION 4, and operation commences after (CR).

Q .... QUIT, returns you to your monitor. You can quit and re-enter as often as you like without affecting the file or ZEN itself.

S .... SORT, will sort and list the symbol table generated during your last assembly. Only the first letter is used to sort but the process is fast and non-destructive.
Upon entry you will be prompted for an output option,these are:

E .... EXTERNAL
V .... VIDEO

Output is generated a page at a time (see ASSEMBLY OUTPUT). You may also choose to display only part of the symbol table by adding a selector key to your command letter.

Example ..... SK(CR)

Would only display symbols beginning with the letter 'K'.

A .... ASSEMBLE, will assemble your source file until the END pseudo-op is encountered. Upon entry you will be prompted for an output option, these are:

E .... EXTERNAL, list to external device.

V .... VIDEO, list to video.

C .... CASSETTE, object file to cassette in INTEL HEX format. You may add an execution address to the command letter and this will be placed in the EOF record of the INTEL dump.

Example C3800H

Places 3800H in the EOF record.The null parameter results in an execution address of zero.

(CR) ....The null option generates no output. This is the fastest mode and should be used until all source errors are eliminated.

X <STRING> DELIM <NEWSTRING> CR

if EOF limited string not found
MEM not enough memory to insert all of string
HUH! syntax error.

V... same as read but doesn't load program into MEM.

NOTE : DELIM = '↑' in XCHANGE STRING.
1000 → 1F60 , SYMBOL AREA 2500 BYTES

POINTER MOVEMENT COMMANDS

T .... TOP, moves the current line to the top of the file, i.e SOF.

B .... BOTTOM, moves pointer to EOF.

D .... DOWN, moves pointer down nn lines, where nn is a numeric paramater.

Example,D37 moves pointer down thirty-seven lines.

The parameter may be between 0 and 65535 decimal. A null parameter (e.g D) or a zero parameter (e.g. D0) will be assigned a default value of one.

Example,D moves pointer down one line.

This applies to all the commands taking a parameter.

U .... UP, moves the pointer up nn lines.

P .... PRINT, displays nn lines on the VDU. The last line displayed is the new current line.

L .... LOCATE, will find an arbitrary string in the file.

Example,LBIT 7,(HL)

moves the pointer to the first line containing the string BIT 7,(HL). If the string is not found then the pointer is at EOF. There are no restrictions on the string content.
The file is searched from the line AFTER the current line. The reason for this is apparent if you consider trying to find all occurrences of a string. Having located the first occurrence that will be your current line. If LOCATE searched from the current line then obviously it would find the string on the same line, forcing you to go DOWN a line every time.

DELAY : (IN SOURCE)

DELAY TOO LONG BETWEEN PRINTING PAGES
CHANGE LOCATIONS 156A, 156B FOR BEST
RESULTS [1569 = LD DE, £1000]

EDITING COMMANDS

Z .... ZAP, erases nn lines from the file, starting from current.

E .... ENTER, enters text into the file. Upon entry ZEN will display the line number as a prompt. Enter a line of text,

Example, START:INC(HL)

and terminate with (CR). The next line number will be displayed and so on. To exit from this command simply key a full stop '.' at the start of a line. You can enter text anywhere in the file. Text is entered at the current line. The old current line, and all following lines, are moved downwards.

N .... NEW, lets you replace an existing line with a new line.

NOTE Althought line numbers are usually displayed you are never required to enter them yourself. They are there solely as a positional guide and are computed dynamically rather than being stored in the file along with the text.

NOTE Should you exceed available memory the error message MEM will be generated. The file will be in a safe state.

347

THE ASSEMBLER

ZEN expects source statements to be constructed according to the syntax defined in the ZILOG Z80 Assembly Language Programming Manual.

Each line of the source file is a statement divided, conceptually, into at most four fields:

```
SETUP:LD A,(STATUS) ;Pickup Drive A state
Label
Operator
Operand(s)
Comment
```

We say conceptually divided because the components of a statement don't have to be specifically positioned into fields. As long as you use the correct seperators (spaces,commas,etc) ZEN accepts statements in free-format.

Comments ..... Comments are ignored by the assembler. They are preceded by a semi-colon ';' and are terminated by end of line.

Operators .... There are 74 generic operators (CALL,LD,JP,etc). In addition there are the pseudo-ops which are described fully later.

Operands ..... The number of operands in a statement depends on the operator. Examples;

```
NOP ................. No operands
CP .................. One operand
BIT ................. Two operands
JR .................. One or two .... (JR SYMBOL)
                                     (JR Z,SYMBOL)
```

Operands may be;

Register names (A,B,HL,IY,etc)
Condition codes (Z,NZ,C,NC,etc)
Numbers

The most complex group is that of the numbers. All the following are evaluated as numbers;

Numbers
Decimal,hex and octal bases are accepted, with decimal being the default base. Hex numbers are 'H' postfixed and octal numbers are 'O' postfixed. Numbers must begin with a digit, a leading zero is sufficient.

ASCII Literals
The assembler will generate the ordinal value of any ASCII character enclosed in single or double quotes.

Symbols
Symbols are explained in detail later.

Location Counter
This is the internal variable which simulates the Program Counter. It is accessed by using $ (dollar).

Operands(cont) .. In addition all of the above data types may be elements of an EXPRESSION formed using the math operators;

+ Addition
- Subtraction
* Multiplication
/ Division
& Logical AND
. Logical OR

An expression may be used anywhere that a simple number can be used. The following are all valid statements;

```
LD HL,START*2-274
LD A,'P'.80H
CP'L'+12
JR NC,$-8
```

Expressions are evaluated strictly left to right with no precedence ordering. Arithmetic is unsigned 16bit integer and overflow will be ignored. Elements in an expression need not be delimited by seperators as the math operators are implied seperators.

Labels ..... A label is a way of marking a statement. Each time you use a JP,CALL,etc you need a way of specifying the destination as an operand. Assembly language allows you to use a symbolic name as a label. BASIC is an example of a language without this facility. In a BASIC program the line numbers are used as labels, e.g. GOTO 170.
Symbols will be explained in greater detail.

NOTE

RLD (HL) for this assembler.
RRD (HL)

SYMBOLS

Symbols are one of the most important features of Assembly Language. A symbol is a name with an associated value, the name is used rather than explicitly stating the value. A symbols value is declared to the assembler in one of two ways;

1 .... By using the EQU pseudo-op. This allows you to assign your own value to a symbol.

Example, CRT:EQU 0C4AH

Every time this symbol is used the correct object code is generated by the assembler.

Example, CALL CRT would produce CD 4A 0C in hex.

A symbol allows easy modification. It is only necessary to change a symbols value rather than changing an operand every time it is used.

2 .... By placing it at the start of a statement. The assembler will assign the value of the location counter to the symbol. The symbol is being used as a label.

Whichever method is used a symbol must be postfixed with a colon ':' when declared. A symbol must begin with a letter but may contain letters or numbers after that. Letters may be upper or lower case. ZEN allows symbols of any length although symbols longer than seven characters will affect the listing. The width of the listing symbol field may be changed by altering one byte (see ZEN Listing).

There are certain reserved keywords which cannot be used as symbols. These are;

Operator names
Register names
Condition codes

Any attempt to use these will result in the error message RSVD. Note that these keywords are all uppercase, using the same name in lowercase would be perfectly acceptable.

PSEUDO-OPS

These are additional operators which have no direct equivalent in the Z80 Instruction Set but are understood by the assembler. They are used in the same way as normal operators.

END	End Assembly	(No operand)
DS	Define Storage	(One operand)
DW	Define Word	(One operand)
DB	Define Byte(s)	(Variable operands)
EQU	Equate	(One operand)
ORG	Origin	(One operand)
LOAD	Load memory	(One operand)

END .... This operator must be used to terminate assembly. Failure to do so will result in an error message and an incomplete assembly.

DS ..... Skips a number of object locations, ie leaves a gap in the object code.

Example DS 47

would skip forty-seven bytes. Commonly used to reserve space for a text buffer,stack,etc., where memory contents don't need to be defined.

DW ..... Generates a word (two bytes) in the object file in reversed order as required by the Z80 16 bit load instructions.

Example BUFFER:DW VALUE

would initialise location BUFFER to VALUE.

DB ..... Generates the value of the operand(s) directly in the object file. Takes as many operands as desired, seperated by commas.

Example DB 47H,'T'*2,32,NEWLINE

Each operand may be an expression but obviously no expression can have a value greater than one byte (255).
The location counter will be incremented after every operand as if each operand were on a seperate line.
In addition to the normal data types any operand may be of a new data type, the ASCII literal string.

Example MESSAGE:DB'STICKY FINGERS',NEWLINE

Strings may be of any length, but, unlike a single character ASCII literal, cannot form part of an expression. A string is formed in the same way as a single literal, by enclosing in matching quotes. Note that single and double quotes are implied seperators like the infix math operators. You may use a quote,of either type, as a literal by using the OPPOSITE type of quote as the delimiters.

PSEUDO-OPS (cont.)

EQU .... Assigns a value to a symbol.

Example NEWLINE:EQU 31

The operand may, as usual, be an expression but there is a restriction on what you may use in the expression. This is because the operand must be capable of immediate resolution. The value of any symbols used in the expression must already be known to the assembler, forward referenced symbols will result in the UNDEF error message.

Example BACKSPACE:EQU NEWLINE-2
NEWLINE:EQU BACKSPACE+2

This sequence is ILLEGAL because each symbol is defined in terms of the other. ZEN would never arrive at the correct value of either symbol. The 'no forward reference' rule is designed to prevent you ever making such a mistake.
In practise you will probably never encounter such a situation as most EQUATES have fairly simple operands.

ORG .... Defines the Origin, or assembly address, of the object code.

Example ORG 10A7H

This operator may be used as often as desired throughout an assembly, to produce sections of code at different locations. The same restriction applies to the operand as that governing EQU operands.

LOAD ... Lets you load the object code straight into memory as it is produced.

Example LOAD 4000H

would load the code commencing at that address, irrespective of the actual assembly Origin. The loading process is entirely independant of the output option specified upon entry to the assembler.
If this operator is not used then no location outside ZEN will be altered in any way.
Note that use of a subsequent ORG operator turns the loading process off, each time you set a new ORG you must specifically re-establish the load process.

OPT 'V' if used will switch on VDU when init on second pass of assembler. useful when used with fast assembling.

ASSEMBLER ERROR HANDLING

If the assembler finds an error in the source code the following will happen;

Assembly terminates.
An error message is displayed.
The incorrect line becomes the editor current line.
The line is displayed.
The command loop is re-entered.

You may now correct the error and assemble again. It is impossible to make an error which will cause ZEN to be damaged. Assembly runs at about four thousand lines a minute so, unless the file is very large, errors are found almost immediately.

ERROR MESSAGES

DBL.SYMBOL ... You have declared the same symbol more than once.

UNDEF ..... You have used an undefined symbol.

SYMBOL ..... You have forgotten the symbol.
Example, EQU 4

RSVD ..... You have used a reserved keyword for a symbol.
Example, PUSH:EQU 4

FULL ..... The symbol table is full (see later).

EOF ..... You have forgotten END and have hit EOF.

ORG ..... No assembly origin

HUH? ..... The assembler doesn't recognise the operator.

OPND ..... Something wrong with an operand.
Examples, LD A,256
BIT 9,B
LD(DE),C

Also included are relative jumps which exceed the allowable range and trying to index out of range.
Example, LD B,(IX-186)

The assembler will catch ALL incorrect statements.

ASSEMBLY OUTPUT

Listing .... ZEN supports two list devices, the normal system console and an External device (usually a printer).
The external driver is NOP'ed out. If you have a printer then you should insert your own driver here.
ZEN generates a CR,LF for a new line and a FormFeed for a new page.

Listings are generated a page at a time, a page consists of sixty lines. Listing can be halted at the end of each page by pressing any key on the console keyboard. Listing will restart when another key is pressed.
The end of page routine DELAY is used to effect this hold facility. DELAY will sample the keyboard status at each pass through it's outer loop. You should insert your own routine to sample status in DELAY. If a keypress exists you should return immediately with the CARRY flag set. The PAGE routine will then call CHARIN.

Object ..... Cassette output is straightforward. The standard vectors SRLIN,SRLOUT are used

SYSTEM OUTPUT

There are four I/O vectors you may need to modify.

CHARIN,SRLIN return a character in register A from the system console and cassette respectively. No other registers may be altered.

CHAROUT,SRLOUT pass a character to the console and cassette respectively. Characters are passed in register A and in register C (for ZAPPLE).

Each vector occurrs only once. ZEN drives the console with a CR,LF for new line and a FormFeed for screen clear. There is space in the Video driver for you to change these if necessary.

The only other control character used by ZEN is backspace. If your console won't respond to a BS then you need only change one byte. This byte is a NOP in routine USER, it should be changed to a LD A,(HL) ie 7EH. The effect of this will be to echo the character just deleted rather than a BS in response to a user backspace.

ODDS and ENDS

Interrupts ..... ZEN will run under continuous or intermittent interrupts. The alternate registers are never used and there are about ten bytes of free stack at maximum stack depth.

Symbol table ..... The ST is the area where each symbol is stored together with it's value. The ST is built during the first assembly pass only.
The ST is positioned above ZEN and below the source file. Sooner or later you'll need a larger ST, here's what to do:

CLOSE the file.
QUIT to monitor.
Locate the SOF and EOF.
Change BOTH pointers to position the file higher up.
Save the new version on tape.

Saving ZEN ..... When you alter ZEN and save on tape you should be aware of two things:

1 The file must ALWAYS be CLOSED.
2 You must save up to the VERY last byte shown in the listing.

Exit ..... The final thing you must alter to customise ZEN for your own system is the exit vector. This is simply the JP to your monitor.