NAS-SEMBLER

A DISC-BASED EDITOR/ASSEMBLER PACKAGE
FOR NASCOM MICROCOMPUTERS WITH
THE NAS-DOS DISC OPERATING SYSTEM

Rev 1.2 15 July 1982

CONTENTS

1. INTRODUCTION

NAS-SEMBLER is a powerful disc-based editor/assembler designed to give the Nascom microcomputer user the facilities normally found on much larger and more expensive assemblers. It is intended for use on any Nascom 1, 2 or 3 computer equipped with discs and the NAS-DOS disc operating system.

NAS-SEMBLER consists of separate edit and assembler modules which can access common source, list and symbol table files. The editor is used to prepare source programs in assembly language using the standard Z80 instruction set. These programs can then stored on disc. The assembler is used to process the source files to generate Z80 machine code in the form of an object file. The object file may then be loaded into memory and executed in the usual way.

In addition to the Z80 assembler module included in this standard system there are also 6800 and 6502 cross assemblers available. These share the edit module, and when used with the Nascom enable users to assemble programs for the 6800 and 6502 microprocessors (a process known as 'cross-assembly'). Cross-assemblers for other microprocessors are under development, including the 16-bit Motorola 68000.

NAS-SEMBLER is provided with translation facilities built into the edit module which provide compatibility with the simpler ZEAP editor/asssembler.

This manual describes the use of the NAS-SEMBLER editor assembler package. It does not set out to teach programming in assembly language. For further information on programming in assembly language the reader is referred to the books listed in Appendix 1.

2. HARDWARE REQUIREMENTS AND PROGRAM CONFIGURATION

If your copy of NAS-SEMBLER has been configured for your system by your dealer you can ignore this section of the manual. Move on to the next section, which provides a simple example of the use of the editor and assembler features of the system.

2.1 Hardware

The minimum hardware requirements for use of the NAS-SEMBLER programs are as follows:

Nascom 1, 2 or 3
16K RAM memory
Nascom single disc drive
NAS-SYS 1 or NAS-SYS 3 monitor
NAS-DOS disc operating system

The system may optionally contain a hard copy printer.

2.2 NAS-DOS location and memory available

As supplied NAS-SEMBLER is intended for use with a standard NAS-DOS stored in EPROM at location D000 hex. If the NAS-DOS is located in any other position the following changes should be made:

1. Editor

Modify locations 101E - 1020 to jump to the start of the NAS-DOS

Modify locations 1021 - 1023 to jump to the DOS
Modify locations 2641 - 2643 to jump to the DOS ARH

2. Z80 assembler

Modify locations 101E - 1020 to jump to the start of the NAS-DOS

Modify locations 1021 - 1023 to jump to the DOS
Modify locations 3287 - 3289 to jump to the DOS ARH

If you are using NAS-DOS in RAM you must make one other change to both the editor and the assembler to avoid the routine which normally finds the top of RAM memory over-writing NAS-DOS itself. In the case of the editor this means modifying locations 10D9 - 10DB to 21 ll hh, where hhll is the address of the top of memory to be used by the editor. The same bytes should be inserted in the assembler, but at locations 10DB - 10DD rather than at 10D9.

You can make these modifications by loading the programs into memory (one at a time of course!) using the NAS-DOS]L command and then using the NAS-SYS command M to modify the locations. The programs can then be saved by executing them at 1000 (E1000) and then typing COPY.

2.3 Printer support

NAS-SEMBLER is supplied with a built-in software driver for a printer connected via the serial (RS232) interface and using a printer 'busy' signal. This driver will suit most serial printers, but in case changes are required (eg for printers using the linefeed rather than carriage return for a newline) the source of this driver is contained in Appendix 3 .

If you are using a printer with a parallel (Centronics) type of interface this should be connected to the Nascom using the information contained in Application Notes AN-0005 and AN-0006. The software for this type of printer can be incorporated by loading the file OPPRINT* from the disc supplied. A listing of this driver is included in Appendix 2, and the source is stored on disc in the file TPPRINT. You can incorporate this permanently in the editor or assembler by saving the program after loading the driver. You can retrieve the original serial driver by loading the file OHPRINT*.

The software for controlling the printer is located at 102A hex in both the editor and assembler. If you require some alternative software to support another printer you can insert your own routines at this point - there are 90 (decimal) bytes available. If you wish to incorporate this alternative driver permanently in NAS-SEMBLER you can do this by saving NAS-SEMBLER on disc after loading the driver routine. You can see how the printer driver functions by referring to the listings in appendix 3.

Remember that if you save a driver on disc and then load this into memory a whole sector (256 bytes) will be loaded. If you load this directly over the editor or assembler you will over-write part of that program! You should therefore load the driver, copy it to empty memory, load the program and then copy the driver back into the program. The files OPPRINT* and OHPRINT* have been saved with the relevant parts of the programs, so this precaution is not necessary in the case of these routines.

*** NB**

Files OPPRINTE and OHPRINTE are intended for use with the editor.

Files OPPRINTZ and OHPRINTZ are intended for use with the Z80 assembler.

3. HOW TO GET STARTED

NAS-SEMBLER is a very powerful package, which by definition means that it has a very wide range of operating commands. It will take you some time to get to know how to exploit all these to best advantage. There is no susbstitute for careful study of the manual in order to fully undrstand the system, but to help you to understand the basic facilities to start with we will work through a simple example.

Suppose we take the example of a program to perform hex to ASCII conversion, as described on page 7-2 of Leventhal's 'Z80 assembly language programming' (Osborne/McGraw Hill, Ref. 1). This program consists of the following (modified to suit Nascom):

```
LD A,(2000H) ; Get hex digit
ADD A,90H    ; Develop extra 6 and carry
DAA
ADC A,40H    ; Add in carry, ASCII offset
DAA
LD (2001H),A ; Store ASCII digit
SCAL MRET    ; Return to NAS-SYS
```

First we must create this text in memory, and then a disc file. To do this insert your NAS-SEMBLER disc in drive 0 and type

```
]E:EDIT
```

The editor will be loaded from disc and will be executed. When the Ok prompt appears you could type

```
NEW
```

(reply YES when asked if you are sure) in order to make sure there is no data present. This is not really necessary when cold starting the editor, but serves as an example of how you could clear the data area in future when there is a file present.

NEW is a 'command', which we can issue since the editor is currently in command mode. An example of another command is DIR - try typing this, and you will get a directory listing for the current disc drive (drive 0) in a similar way to the]D command of NAS-DOS.

Returning to our example, we have already typed in most of this for you, and it is contained in the file TDEMO. To load this ready for editing you should type

```
LOAD TDEMO
```

To enter text in an empty data area, to insert text in the current file or to edit the current file, we need to use the 'edit' mode. We can do this by typing the command EDIT. You should now be in edit mode, and you can now proceed to view, complete and correct the demonstration program. Now that you are in edit mode you should have the program shown in figure 1 in memory. The first 15 lines will be displayed. If you use the cursor down key you can cause the screen to be scrolled to reveal the remainder of the program. The cursor up key can be used to scroll back to the start, or by pressing CTRL/G you can return immediately to the beginning of the program.

If you look at the listing you will see that we have left off the last SCAL MRET instruction. You can add this to the end of the data by moving the cursor to after the last line (cursor down, or more quickly type CTRL/J). Now type in the missing line - but type a space in the first column to indicate that there is no label. The editor will automatically tab to column 7 to retain a tidy listing. Now type in the SCAL MRET and press the ENTER key.

Next you will see that we have ommitted the first DAA statement. You can insert a line by placing the cursor at the start of the line before which you wish to insert a line and pressing CTRL/I. The editor will insert a line containing only the ';' symbol - if you press ENTER you will have inserted a blank comment line. In our case we should press the space bar, to clear the label field, another space to tab to the operand field and then type in the DAA instruction, followed by the ENTER key.

The next mistake is in the first line of the program where the address is typed as 40H rather than 2000H (we copied the example too closely!). To edit this line we place the cursor on that line and use the NAS-SYS editor in the normal way to correct the mistake. You MUST press ENTER to effect the change.

The final mistake is the rubbish contained in the third comment line - let's just delete the line. To do this place the cursor on that line and type CTRL/K.

We have now completed the program, and we wish to save it on disc. First we must terminate edit mode and return to command mode. To do this use the ESC key (shifted ENTER). To save the file type

SAVE TDEMO2

We could have used the original name TDEMO, which would have resulted in the usual NAS-DOS 'Overwrite - sure' prompt.

Now that the editing is complete we wish to assemble the program. You can call the assembler direct from the editor by typing

/Z80

Most assemblies will be done in MODE 1 (save in memory image), and this is the default assembly mode. Now we need to specify the output required. We will simply list to the screen, so type O1. The assembler will default to 54 lines per page to suit a printer with 66 line per page fanfold paper. So that we can view the output on the screen we should type LINES 18. Now we can request the assembler to assemble the program - to do this type

P2 TDEMO2

Actually as we have just been editing this file we could have ommitted the file name and just typed

P2

The assembler will now proceed to assemble the program and list it. The listing will pause after the screen fills; pressing any key will resume listing. The ESC key can be used at any time to terminate the assembly. If you made any mistakes in correcting the program you may have errors. You can invoke the editor again simply by typing EDIT. The editor will be loaded automatically - press the ENTER key in response to the Edit? prompt to go into edit mode (the current text file is still in memory, and you need not on this occassion re-LOAD it). Correct any mistakes and then use the /Z80 command to try assembling again.

You now should have an understanding of the principles of NAS-SEMBLER. We suggest you read the remainder of the manual to find out all the facilities available to you using the NAS-SEMBLER editor and assembler.

4. GENERAL FEATURES

The editor may be executed by typing

```
]E:EDIT
```

The assembler may be executed by typing

```
]E:Z80
```

Either can be cold-started if already present in memory by executing at 1000 hex, or it may be warm-started by executing at address 1003 hex.

A number of features are common to both the editor and assembler programs, and these are described in this section of the manual.

4.1 General commands

COPY
This causes a copy of the current (EDIT or Z80) program to be saved on the current disc.

DOS
This command cause a return to cold start NAS-DOS.

LINES
Sets the number of lines to be printed on each page. Set at 54 for 66 line fanfold paper as the default. For screen output use the command LINES 18 (the space is obligatory in the assembler). Note that listing of assembly pass 2 listings (L files) from disc will use the LINES command value active when the assembly was performed, rather than the current value.

LIST Lfile n
This provides a listing on the display of a pass 2 listing (Lfile) file on the current disc. If n is specified then the listing begins at this page of the listing. Note that the first page of the listing is page 0!

LLIST Lfile n
Performs the same operation as LIST, but output is directed to the printer.

MON
This causes a return to NAS-SYS.

NEW
This command causes the text and symbol table areas to be cleared. You will be prompted with "Sure?" to avoid accidents - type Y if you wish to proceed with the clearing operation.

OLD
Causes the previously used text file to be made accessible again in memory. This can be used to cancel a NEW command, but since it is a command based on memory (not disc) data it is only effective if the text area has not been over-written in memory. Note that you should only use this command if a text file is present in memory.

?
This command displays the amount of memory space remaining (in decimal).

/Z80
This command causes the editor program to terminate and control to be transferred direct to the Z80 assembler.

4.2 Disc Commands and file names

The NAS-SEMBLER editor and assembler contain their own commands for performing disc access operations. The normal NAS-DOS commands prefixed by] are disabled while in NAS-SEMBLER.

NAS-SEMBLER uses an obligatory file naming convention to distinguish between different types of file. This consists of a prefix letter specifying the file type as follows:-

L	Listing file from pass 2 of assembler
O	Object code file
S	Symbol tables
T	Text (source) file

All files accessed by the NAS-SEMBLER system must conform to these conventions.

The disc access commands provided are:-

DISC n
Select disc number n for future disc access

DIR
Display the disc directory on the screen

LDIR
Display the disc directory on the printer

LOAD Sfile Tfile1 Tfile2 ....
Loads the files into memoryfrom the currently selected disc drive. The files may be of type source and/or a single symbol file. Note that source text files will be appended to the end of any current source file so that the command NEW should be used first if required.

SAVE Sfile Tfile
Save the current source and/or symbol file

DELETE file1 file2
Scratch the specified files from the disc directory

/file
If the file name begins with O then the file is treated as an object file and loaded into the appropriate location in memory using the object loader (see MODE in the assembler manual). If this would involve overwriting the editor itself the message "Can't" will be displayed and the editor will be aborted. When loading is complete the program waits for a key to be pressed; if this is ESCape the program returns to NAS-SYS, otherwise a jump to 1000 hex is performed. If the filename does

not begin with T, L or S it will be loaded and executed as a normal machine code file.

/Z80
As described above this command will cause the Z80 assembler to be invoked from within the editor.

5. The NAS-SEMBLER editor

5.1 Edit commands

All commands issued from the editor, including those described in section 4, can be abbreviated to the first two letters and a full stop.

A number of edit commands make use of line numbers. Whilst the lines themselves contain no line numbers the line window at the top of the screen indicates the line number of the first line of test displayed on the screen. You may therefore note when in edit mode the lines to be operated upon before returning to command level and issuing the appropriate command.

DELETE a b
Delete line numbers a to b in the text.

DELETE S
Deletes any symbol table in memory

EDIT
Enter text edit mode, and display one page of text starting from the topmost line (see 5.2 below).

n EDIT
Enter edit mode and display one page of text starting at line n.

EDIT string
Enter edit mode and display one page of text starting from the first line containing the character string specified.

n EDIT string
Enter edit mode and display one page of text starting from the line containing the n'th occurrence of the string specified.

INSERT a b
This will cause 'a' blank lines to be inserted before line number 'b' in the text.

n PUT a b
Transfer line numbers between 'a' (start) and 'b' (end) to a position immediately before the line number 'n'. The target line 'n' must not be between 'a' and 'b'.

5.2 Text editing

Source text files for the Z80 are arranged in four columns - label, opcode, operand and comment.
Labels should start at the beginning of a line and start with an alphabetic character (A - Z). The remaining characters in the label field can be A - Z or 0 - 9. The label must not exceed 6 characters in length. Since the label cannot contain a space, pressing the space bar will automatically cause the editor to tab to the opcoded field. Similarly pressing the space bar in the opcode field will automatically cause a tab to the operand field. This effect is automatically

cancelled if the character ';' has been used in column 1 to indicate that the whole line is a comment. Note that the ';' is only recognised in the label or opcode field if it is in the first column of that field.

Empty lines are not permitted. You may, however, enter 'blank comment' lines, ie lines containing only ';' in column 1.

No line may exceed the length of a line on the display. Attempts to type beyond this point will be inhibited. Note that ZEAP files to be translated should be truncated where necessary to avoid contravening this condition BEFORE translation is attempted (see appendix 3).

The following special key codes are used in edit mode to control editing operations:

SHIFT/BS or CTRL/L or GRAPH/Q
Alternately locks and unlocks the shift key (ie toggles between upper and lower case).

↑ ↓ ← → SHIFT/← SHIFT/→ BACK LF CH
All the cursor control keys function normally. Note that the cursor up/down keys also control scrolling.

CTRL/A
Delete all text After the line on which the cursor is located. For safety this command must be typed twice.

CTRL/B
Delete all text Before the line on which the cursor is located. For safety this command must be typed twice.

CTRL/C
Continue search for string specified on entry to the edit when using the EDIT string construction. If no further occurrence of the string is found the command is ignored, otherwise the screen is adjusted so that the line containing the string is at the top.

CTRL/D
Duplicate the current line.

CTRL/G
Go to the start of the text file.

CTRL/I
Insert a blank comment before current line. This can be used to enter any new lines in the middle of the text file by over-typing the comment lines.

CTRL/J
Jump to the end of the text file.

CTRL/K
Kill the current line.

CTRL/N
Display Next page of text file.

CTRL/P
Display Previous page of text file.

CTRL/X
Send text to printer starting at the top of the current display.

5.3 Miscellaneous commands

ASCII
In this mode the codes of any keys typed are displayed. The command is terminated by pressing the <ENTER> key (code 13 decimal, 0D hexadecimal) .

CONSTRUCT B n aaaa
It is sometimes necessary to have tables of data in a source file. These tables may be difficult to create in the assembler itself - for example a set of values in logarithmic sequence. The CONSTRUCT command allows the user of poke the necessary values into memory, using for example BASIC, and then retrieve these using the assembler. The command produces n DEFB pseudo op's followed by a byte peeked from memory starting at the hexadecimal address aaaa. The value n should be expressed in decimal. The DEFB table will be added to the end of any source text file already in memory.

CONSTRUCT W n aaaa
This is identical to the CONSTRUCT B command, except that DEFW pseudo op's are used and are followed by 16 bit values. This could be used for compiling tables of addresses, for example.

D (expression)
Causes the decimal result of the given arithmetic expression to be displayed. Any values in the expression preceded by 0 are treated as hexadecimal.

H (expression)
Identical to the D command, except that the result will be displayed in decimal. Any values in the expression preceded by a 0 are treated as hexadecimal.

MAP
The display memory map is displayed with a look up table showing the first 128 characters from the character set. Pressing any key except <ESCAPE> will cause the second 128 characters to be displayed.

5.4 Error messages

Arithmetic overflow
A number has exceeded signed 16 bit resolution.

Can't find
May be caused by referencing a disc file which is not on the disc, or by an attempted search for a non-existent string in the text.

End of memory
Buffer space exceeded.

Name?
Caused by any improper, missing or un-necessary name. Also caused by an incorrect letter at the start of a file name.

Syntax error
Caused by any incorrect characters, missing brackets etc in an expression used in the D or H commands.

What?
Caused by mis-spelled or illegal commands, or by improper, un-necessary or missing arguments in commands.

5.5 NAS-SEMBLER EDITOR COMMANDS - A SUMMARY

ASCII	Print ASCII values
CONSTRUCT B n aaaa	Construct data file from memory 'peeks'
CONSTRUCT W n aaaa	Construct data file from memory 'deeks'
COPY	Save copy of EDIT on disc
CTZ	Convert NAS-SEMBLER file to ZEAP (mem>mem)
CTZ Tfile	Convert NAS-SEMBLER file to ZEAP (disc>mem)
CTZ zfile	Convert NAS-SEMBLER file to ZEAP (mem>disc)
CTZ Tfile zfile	Convert NAS-SEMBLER file to ZEAP (disc>disc)
D (expression)	Print decimal result of expression
DELETE x y	Delete lines x to y
DELETE S	Delete symbol table
DIR	List disc directory on screen
DISC	Switch disc access to drive 0
DISC n	Switch disc access to drive n
DOS	Cold start NAS-DOS
EDIT	Start edit at line number in line window
n EDIT	Start edit at line n
EDIT xyz	Find first occurence of xyz
n EDIT xyz	Find n'th occurrence of xyz
ERASE file	Scratch file from directory
H (expression)	Print hex result of expression
INSERT x y	Create x blank lines before line y
LDIR	List disc directory to printer
LINES n	Set n lines per page print/display
LIST Lfile	List listing file on screen only
LIST Lfile n	List listing file on screen from page n
LLIST Lfile	List listing file to printer
LLIST Lfile n	List listing file to printer from page n
LOAD Tfile	Load source text file
LOAD Sfile	Load symbol table file
LTAB	List symbol table to printer
LTAB label	List symbol table to printer starting @ label
MAP	VDU map and character set
MON	Return to NAS-SYS monitir
NEW	Clear all text and table from memory
OLD	Restore old text
n PUT x y	Transfer lines x to y into position n
SAVE	Save text file using name in file
SAVE Tfile	Save text file with name Tfile
SAVE Sfile	Save symbols in file Sfile
TAB	List symbol table on screen
TAB label	List symbol table on screen starting @ label
USE aaaa	Specify users printer routine
ZTC	Convert ZEAP file to NAS-SEMBLER (mem>mem)
ZTC zfile	Convert ZEAP file to NAS-SEMBLER (disc>mem)
ZTC zfile Tfile	Convert ZEAP file to NAS-SEMBLER (disc>disc)
?	Print memory size
/Ofile	Load/execute object file
/xfile	Load/execute other file
/Z80	Invoke the Z80 assembler

6. THE NAS-SEMBLER Z80 ASSEMBLER

6.1 General commands

Ctrl/P and Ctrl/O
The control keys CTRL/P and CTRL/O are used in command mode to control the paging of listings. Normally listings are halted after each 'page' of output. Pressing CTRL/O causes continuous listing (although gaps are still left between pages). CTRL/P will restore the pause after each page.

ALPHA
This command will cause any symbol table in memory to be sorted into alphabetic order. Long tables can take an appreciable time to sort (up to a minute).

EDIT
Causes the editor to be re-loaded and control transferred back to it. A prompt Edit? is issued. Pressing the ENTER key will cause edit mode to be invoked - any other key leaves the editor in command mode.

MARGIN n
The width of the left hand margin is set to the value specified for all output to the printer.

6.2 Assembly control commands

The NAS-SEMBLER Z80 assembler is a two pass assembler, the first of the passes being performed in two loops. It can produce absolute (memory image) object files or files which can be loaded using a very simple loader.

Several functions are performed only during one of the passes - the QUERY command is only executed during the first loop of pass 1, and all errors except R/D are detected during the second loop of pass 1.

Note that the output commands (commencing with the letter O) allow a variety of list outputs OR production of object code. You cannot produce both object code and a listing in the same run. To overcome this limitation you simply use the P2 command twice, changing the Output option to produce a listing and object code.

The commands controlling the assembly process are as follows:

O0
Specifies that object code only will be output to disc. The assembler will prompt for the file name - which must conform to the conventions by beginning with the letter O.

O1
An assembly listing will be produced, routed to the display screen only.

O2
The assembly listing will be sent to both the display and printer.

O3
The listing output will be directed to the display screen and to a disc file. You will be prompted for the name of the file, which MUST follow conventions by starting with the letter L.

O4
The listing will be sent to the display, printer and disc. The filename must be specified starting with the letter L when requested.

P1 Tfile1 Tfile2 Tfile3 ....
Performs pass 1 on the file(s) specified. If no file is specified the program currently in memory is assembled. Pass 1 simply performs error checking and compiles a symbol table. Errors will be reported on the display as they are found.

P2 Tfile1 Tfile2 Tfile3 ....
Performs pass 1 exactly as described for P1, and then proceeds to perform pass 2. Output will be directed as specified by the output (O) commands described below.

MODE n
Selects the format of the object file to be produced under output option O. Mode 1 is the default mode, and produces absolute, memory image, files which can be loaded and executed using the normal NAS-DOS]L and]E commands. The sections of source text to be assembled in this way MUST BE PRECEDED BY the ENDS pseudo op and terminated with the SUPP pseudo op. In addition the ENT pseudo op is required before the statement the address of which is to be used as the execution address when the NAS-DOS]E command is used. See figure 1 for an example of the construction of a text file containing these commands. Further information on pseudo ops is included in section 6.3.

Mode 0 produces object code which is loaded using the NAS-SEMBLER loader, as described in section 4.1.

6.3 Opcodes and pseudo ops

The mnemonics used for the opcodes are the standard ones for the Z80. Certain extensions are available as follows:

1. The PUSH, POP, INC and DEC opcodes can have multiple operands - eg PUSH HL,AF,BC,IX... .

2. Certain opcodes can be followed with a single 8 bit argument to indicate the number of operations required. The opcodes involved are:

RDEL EI ROUT DI BRKP EXX RIN STAR HALT CCF SCF DAA RRA RLA RRCA RLCA NOP

eg NOP 4 will insert 4 NOP's.

Note:-

1. Opcodes with relative offsets in which the argument is preceeded with + or - will be assumed to refer to displacement values. Otherwise

the reference is assumed to be to the destination, and the displacement will be calculated automatically.

2. In the 8 bit arithmetic codes CP, AND, OR, SBC, XOR, SUB, ADD, ADC the first operand may optionally be ommitted if it is A.

3. In the IN and OUT opcodes all the operands must be included.

4. In conditional operations the conditional should be entered in the operand column - ie JRNC LABEL is not acceptable.

5. Arithmetic operations in the assembler are performed in signed 16 bit mode, giving a resolution of -32768 to +32767. The priority of the operators is as follows (highest priority first):

( )	Parentheses
-	Negation
* / & ¾ < > =	Mutiply, divide, and, or, xor, greater, less equal

6. Numeric values are specified as follows:

%	Binary numbers should be preceeded by %
0	Numbers preceded by a zero are normally treated as hexadecimal and may optionally be followed by H. The RAD instruction may be used to re-define the meaning of the leading 0.
1..9	Unless qualified by any of the above these numbers are assumed to be decimal. They may optionally end with D.

7. Character literals may be used, represented by enclosing them in " marks (eg "A"). The assembler will translate these into the ASCII values. Graphics characters are permitted.

Pseudo ops

Pseudo ops are instructions contained in the source text file which affect the operation of the assembler, but do not result in any object code being generated in the final program. For example, if we wish the listing of the program to start on a new page at some point we could insert in the source file the instruction ENDP. Obviously this has no effect on the final object code program, but will cause the assembler to jump to the top of a new page during listing.

Conditional pseudo ops

The first group of pseudo ops are conditional, that is they will only be performed if the argument specified is true. If no argument is present they will be performed unconditionally. For example the pseudo op END will cause termination of the pass, while END LAB=0005H will only terminate the pass if the value of LAB is 0005H. These pseudo ops are:

END
Forces the pass to end as though the last line of the file had been reached.
ENDP
Forces a new page in the listing.

NON / NOFF
Switches on or off the line numbers on the listing.

PAUSE
Causes the assembler to wait for a key to be pressed. If the key is ESCAPE the assembly terminates and command mode is entered.

SKIP
Causes a blank line to be inserted in the listing.

Block definition pseudo ops

These pseudo ops bracket a block of source text code. They cannot be nested (ie each block must be closed before another is started).

COND / ENDC
COND is used to cause optional skipping of assembly of part of the source text. Assembly will resume when the next ENDC is encountered. Note that ENDC is also a conditional pseudo op.

```
     COND     LABEL>2
```

would cause assembly to be ceased if LABEL is not greater than 2 until an ENDC is encountered.

FOR / NEXT
This is a repeat macro construction. For example:

```
LABEL  FOR      0CH,0DDH,5
       ......
       NEXT
```

will cause the text between the FOR and NEXT to be assembled as many times as there are arguments. At each loop the next value in the series is assigned to LABEL until all have been covered. Assembly then continues on the line after the NEXT. Note that NEXT is also a conditional pseudo op.

MACR /RPTM
This is also a repeat macro construction, but only one argument is needed:

```
     MACR     n
     ......
     RPTM
```

This will cause the text between the MACR and RPTM to be assembled n + 1 times. RPTM is a conditional pseudo op.

INSERT / ENDI
This is a form of macro instruction in which a text file is retrieved from disc and inserted at the point specified by the INSERT filename instruction. The text file will be read and assembled until an ENDI

instruction is encountered. ENDI is a conditional pseudo op.

```
     LD     SP,01000H
     INSERT TFILE
     LD     A,VALUE
```

would cause the contents of the file TFILE to be assembled between the LD SP and LD A lines.

Note that INSERT can only be used when the pass is being performed from a disc file, and not from memory.

Other pseudo ops

ADDR
Reverses the least significant and most significant bytes of an argument. Several arguments can be specified, separated by commas.

DEFB
Defines a single byte numeric value to be inserted in the object code. Several values may be included on the same line separated by commas. A £ symbol may be used to prefix an argument, in which case only the least significant byte of that argument will be inserted (see example in PRS below).

DEFM
Defines a message string to be inserted in the object code. The string should be enclosed by " marks.

DEFS
Specifies number of bytes of storage space to be reserved within object code.

DEFW
Defines double byte numeric arguments to be inserted in the object code.

EQU
This pseudo op equates a symbol or label to the left of it to the value on the right of it. Symbols in NAS-SEMBLER can be redefined at any time (subject to R/D? messages) EXCEPT when they have been defined by an EQU operation.

DEFL
This performs the same function as EQU, but the label is not protected, and may be redefined.

EXEC
This enables the user to call a machine code routine during the assembly process.

```
     EXEC 0D800H
```

will execute a call to D800. All registers are saved and the stack is set up for 20 levels of pushes/pops.

EXT
Sometimes it is useful to reference a symbol table generated by another

program. The EXT statement causes the specified external symbol table to be added to the one already in memory. Thus

```
     EXT     SFILE
```

would cause the contents of the file SFILE to be added to the symbol file. No checking is performed for multiple definition of symbols.

LET
This is one of a group of pseudo ops which perform arithmetic on symbols. LET is used as follows:

```
     LET     PRICE=VAL/14
```

will set the value of PRICE to VAL/14.

LRRC / LSRL / LRLC / LSLA / NOT
These perform shift operations on the values of the labels, For example

```
     LRRC   LAB1
```

will perform a rotate right circular on the value of LAB1. The NOT function simply inverts the state of all the bits.

LSB
Returns the ~~most~~ LEAST significant byte of the argument. Several argumants may be given, separated by commas.

MSB
Returns the most significant byte of the argument. Several arguments may be specified, separated by commas.

NAME
This sets the name to be inserted in the title block during list output. The name may be up to 29 characters long, and may contain any characters.

OPCD
This allows the user to define names and associated strings of data to be used as opcodes. Thus

```
ULDHL  OPCD     07E,023,066,06F
```

will create an opcode with the name ULDHL which performs the operation LD HL,(HL). Once defined ULDHL may be used as an opcode at any point. Four hundred bytes are reserved for user opcode tables within the assembler. The opcode name MUST begin with U.

PHASE / DEPH
While the ORG statement defines the address at which a program will load, PHASE sets the address at which it is configured to run. Thus

```
     ORG      1000H
     PHASE   0D000H
     ....
```

will produce a program designed to run at D000 but to load in at 1000. This can be useful when developing programs which are to be stored in

ROM. DEPH cancels the effect and restores the assembly to normal.

PRINT
This will cause arguments in the operand field to be printed out during pass 1/a. Thus:

```
     PRINT LAB1,"abc",LAB2
```

will cause the values of LAB1 and LAB2 to be displayed, separated by abc .

PRS
Generates the NAS-SYS RST PRS operation. It is followed by the characters to be output. If these are in the form of a string (rather than the ASCII values) the string should be enclosed in " marks. Several characters or strings may be specified, separated by commas. The string should normally be followed by £0 to denote the end of the strings to be output by the NAS-SYS PRS routine. A £ sign prefixing any of the arguments indicates that only the least significant byte should be inserted.

```
     PRS "ABC",£"A"+080H,£0
```

would be assembled as

```
EF 41 42 43 C1 00
```

QUERY
This will cause the assembler to prompt for a value of the symbol in the operand field during pass 1/a.

```
     QUERY LAB1
```

will prompt "Value for LAB1?" and wait for the user to enter a hex argument, which will then be assigned to LAB1.

RAD
Sets the arithmetic radix to which numbers have a leading zero will be evaluated.

```
     RAD      8
```

will set the radix to 8 - octal. The radix may be set to any value between 2 and 16. Such numbers may end with Q, provided that the radix is not 16. The default radix is 16 (hexadecimal) on start up.

6.4 NAS-SYS access

The following NAS-SYS resets are automatically interpreted by the assembler:

BRKP	**(FFH)**
PRS	**(E7H)**
RCAL	**(D7H)**
RDEL	**(FFH)**
RIN	**(CFH)**
ROUT	**(F7H)**
SCAL	**(DFH)**

STAR (C7H)

PRS, RCAL and SCAL must be followed by an argument.

The NAS-SYS SCAL operations MRET to SCALI (5BH to 7FH) are built in as standard, but can be re-defined by the user.

6.5 Error messages

NAS-SEMBLER distinguishes between three categories of error - non-significant, significant and fatal.

Non -significant errors will be reported but will not stop assembly, significant errors will be reported and assembly will terminate at the end of pass 1 and fatal errors will terminate assembly immediately.

Fatal errors

End of memory
Insufficient memory space.

Name?
Incorrect file name (wrong starting letter).

Can't find
Reference to a file which is not in the directory of the current disc.

End Insert?
Inserts must end with the ENDI pseudo op before the end of the file.

End For?
FOR/NEXT loops must be closed before the end of file.

End Conditional?
Conditional assemblies must be terminated before the end of the file.

End Macro?
MACR/RPTM loops must not run over the end of the file.

Significant errors

L/S
Label syntax error. Label starts with 0 to 9, too long etc.

O/e
Operand error. Wrong, illegal, missing or unnecessary operands. eg LD A,HL - brackets ommitted.

OPC
Opcode error. Illegal or mis-spelled opcode.

A/O
Arithmetic overflow. An argument has exceeded 16-bit resolution.

For?
Attempt to nest FOR/NEXT loop.

Macr?
Attempt to nest MACR/REPT loop.

Insert
Attempt to nest INSERT's.

U/S
User opcode has exceeded space reserved for it.

Non-significant errors

REL
Relative offset (in JR) out of range +-128 bytes.

EQU
Attempted re-definition of an EQU protected symbol.

?/0
Division by zero in arithmetic expression.

R/D
Re-definition of a non-protected label. The message will be issued whenever such an action occurs. If you wish to ignore it in FOR/NEXT loops you may use CTRL/O to switch off paging.

O/P
ORG change without de-phase.

C/F
Can't find. A symbol has been referred to which is not in the symbol table. The value will be set to zero.

6.6 NAS-SEMBLER Z80 ASSEMBLER COMMANDS - A SUMMARY

ALPHA	Alphabetic sort the symbol table
COPY	Copy assembler to disc
CS	Clear symbol table
DIR	List disc directory on screen
DISC	Switch disc access to drive 0
DISC n	Switch disc access to drive n
DOS	Return to cold start NAS-DOS
EDIT	Load NAS-SEMBLER editor from disc
LDIR	List disc directory to printer
LINES n	Set output for n lines per page
LIST Lfile	List listing file to screen only
LIST Lfile n	List listing file to screen
LLIST Lfile	List listing file to printer
LLIST Lfile n	List listing file to printer from page n
LTAB	List symbol table to printer
LTAB label	List symbol table starting at label
MARGIN n	Set printer left margin width
MON	Return to NAS-SYS
NEW	Clear text and symbol table
O0	Object code to disc
O1	Assembly listing to screen
O2	Assembly listing to screen and printer
O3	Assembly listing to screen and disc
O4	Assembly listing to screen, print & disc
OLD	Restore old text
P1	Assemble pass 1 only
P2	Assembly pass 1 and 2
SAVE Sfile	Write sysmbol table to disc
TAB	List symbol table to screen
TAB label	List symbol table starting @ label
USE nnnn	Switch printer drive software
?	Print memory size
/Ofile	Load/execute mode 0 object file
MODE 0	Normal linked object output
MODE 1	Pure object output

APPENDIX 1 - References

1. Leventhal, L A, "Z80 Assembly language programming", Osborne/McGraw Hill, 1979

2. Zaks, R, "Programming the Z80", Sybex

3. Wilson, G, "Machine code programming for the Nascom 1 and 2", Interface Components

APPENDIX 2 - ZEAP/NAS-SEMBLER compatibility

Both ZEAP and NAS-SEMBLER use standard Z80 assembler language mnemonics in the source program text, and produce identical object code from that source when assembly is carried out. However, the format in which the source text is stored in memory and disc is different. This means that ZEAP files cannot be read by the NAS-SEMBLER editor or assembled by the NAS-SEMBLER assembler. Conversely the source files created using EDIT cannot be accessed by ZEAP.

To allow existing ZEAP files to be used with NAS-SEMBLER the latter contains an integral translator. This can also be used to convert NAS-SEMBLER text files into a form suitable for use by ZEAP. The translator can generally process correctly over 90% of source files. The following limitations exist:

Conversion from ZEAP to NAS-SEMBLER

1. Hex constants should be prefixed by the £ symbol.

2. Long comments (where the total length of the NAS-SEMBLER source text, including the expanded tabs, exceeds characters) will be truncated by the translator.

Conversion from NAS-SEMBLER to ZEAP

The extended features of NAS-SEMBLER, such as instructions of the type PUSH AF,HL,DE cannot be translated into ZEAP. Such statements should be converted to standard mnemonics either before or after translation so that they can be assembled using ZEAP.

Translation options

ZTC
This converts a ZEAP file to NAS-SEMBLER format, both being in memory, starting at 3A00 hex.

ZTC zeapfile
Converts a ZEAP source file on disc into a NAS-SEMBLER source in memory.

ZTC zeapfile Tfile
Converts a ZEAP source file on disc into a NAS-SEMBLER text disc file with the name Tfile.

CTZ
A NAS-SEMBLER text file in memory is converted into a ZEAP source format in memory. The ZEAP file will be located in memory starting at 3A00, and the address of the end of the file will be contained in 35FC.

CTZ Tfile
The NAS-SEMBLER disc text file Tfile will be converted into ZEAP format and left in memory, in the locations specified above.

CTZ zeapfile
The NAS-SEMBLER text in memory will be converted to ZEAP format and stored on disc with the name 'zeapfile'.

CTZ Tfile zeapfile
A NAS-SEMBLER file is converted to ZEAP format and stored on disc.

APPENDIX 3 - Printer driver routines

```
NAS-SEM - Nascom Z80 assembler  Page 0000


                         0000 ;SERIAL PRINTER DRIVER FOR CREATOR & NAS-CAL
                         0001 ;-------------------------------------------
                         0002 ;
                         0003 ;REV 1.4     6 JULY 1982
                         0004 ;
                         0005 ;COPYRIGHT (C)1982 LUCAS LOGIC LIMITED
                         0006 ;
                         0007         ORG    01029H
                         0008 ;
                         0009         ENDS
                         0010 ;
                         0011 ;PRINTER INITIALISER
                         0012 ;
1029 E5                  0013 HANDS   PUSH   HL ;SAVE REGISTER
102A 21 34 10            0014         LD     HL,HANDR ;SET USER ROUTINE IN
102D 22 78 0C            0015         LD     (0C78H),HL
1030 DF 55               0016         SCAL   "U" ;ENABLE PRINTER ROUTINE
1032 E1                  0017         POP    HL ;RESTORE REGISTER
1033 C9                  0018         RET
                         0019 ;
                         0020 ;CHARACTER OUTPUT ROUTINE
                         0021 ;
1034 F5                  0022 HANDR   PUSH   AF ;SAVE CHARACTER
1035 DB 00               0023 HERE    IN     A,(0) ;WAIT TILL FREE
1037 E6 80               0024         AND    080H
1039 28 FA               0025         JR     Z,HERE
                         0026 ;
103B F1                  0027         POP    AF
103C DF 6E               0028         SCAL   06EH ;PRINT IT
103E C3 05 D4            0029 EXIT    JP     0D405H ;RETURN TO DOS
                         0030         SUPP
```