GENER-80 Z80 ASSEMBLER

OPERATING MANUAL

NASCOM VERSION 1.1

SEVEN STARS PUBLISHING

Published by Seven Stars Publishing, 15 Gloucester Avenue, London NW1 7AU.

First edition 1983

Printed in Great Britain

1. INTRODUCTION

Gener-80 is an editor-assembler for Z80 microcomputers. It enables **the user to create a source program consisting of** Z80 **assembly-language instructions and assembler directives, to edit this program, and finally to assemble it into a machine-code object program resident in memory.** Facilities are available to save the **source program on tape or print it for later reference.**

This manual gives a complete guide to installing and using Gener-80 on your Nascom system. It does, however, assume that you **are familiar with the basic principles and terminology of Z80 assembly-language programming.** Although the manuals supplied by **Nascom give an introduction to the use of machine code, the newcomer is advised to consult one of the many books on the market which give a more complete view of the subject, such as** *Z80 Assembly Language Programming* by Lance Leventhal (published by Osborne & Associates Inc).

Any manual tends to make the operation of a system sound more complicated than it really is, and this one is no exception! We therefore suggest that, after reading it through and installing Gener-80, you enter a short program of a dozen lines or so and try out each command before attempting any serious programming.

2. INSTALLATION

Gener-80 occupies 6.75k of user ram (1000-2AFF hex) and requires a further 256 bytes of workspace (0F00-0FFF hex). The minimum system therefore consists of a Nascom 2 (or Nascom 1 with Nas-Sys monitor and Cottis-Blandford tape interface) with at least 8k of ram commencing at 1000 hex, a tv or monitor, and a cassette tape recorder. However, a printer is almost essential for serious work so that you can produce proper listings of your programs for later reference. As supplied, Gener-80 supports serial printers but Appendix 2 shows how you can redirect its print routine to your own routines, eg if you have a parallel printer.

Gener-80 is supplied on tape cassette as two copies: one at 1200 baud, the other at 300 baud. Both are in standard Nascom 2 format. The "R" command of Nas-Sys should be used to load the 1200 baud copy. Gener-80 should then load from 1000 to 2AFF hex. If there are errors present, you will need to experiment with the volume control on your tape recorder. The level can be quite critical. Start with it low and increase it a small amount after each attempt until best results are obtained. A rough idea of the correct level can be gained by listening to a tape that you have recorded previously. If you still have problems, try setting your tape interface to 300 baud and loading the second copy.

While you may find it difficult to get a completely error-free loading, there may only be, say, two or three corrupted blocks each time. If this is the case, keep trying until you get the minimum number of errors. Make a note of which blocks are corrupted, then save this copy to another tape. Now keep reloading Gener-80 again until these blocks are error-free. Copy this version into a free part of your memory using the "I" command of Nas-Sys, then reload the first version. You should now be able to create an uncorrupted version by carefully copying good blocks from the second version into the first.

When you believe that Gener-80 is correctly loaded into memory, enter E1000 and Gener-80's sign-on message should be displayed (see

ection 4). **It will also check itself for data errors - if it finds any it will display SYS ERROR at the top right of the screen and return to Nas-Sys. If no such error message is obtained, return to Nas-Sys by pressing RESET and save this copy to another tape by entering W1000 2B00. If it does show SYS ERROR and you are sure that you loaded it correctly you may have a hardware fault on your computer.**

3. GENERAL POINTS

3.1. Data entry

A similar routine is used for all user data and program entry, although some extra cursor control functions are available in edit mode. A blinking cursor is provided with the following control commands (the @ key can be used as the CTRL key on the Nascom 1):

Function	**Key**
Backspace	BACKSPACE
Cursor left	Left arrow or CTRL/Q
Cursor right	Right arrow or CTRL/R
Cursor to end of line and enter	CTRL/E
Enter line	ENTER (NEWLINE on Nascom 1)
Delete character	SHIFT/Left arrow or CTRL/U
Insert character	SHIFT/Right arrow or CTRL/V
Escape	ESC or SHIFT/ENTER

Cursor right does not operate beyond the current length of the line. The maximum number of characters in a line is 47, the 48th always being blank. Overflow on to following lines is not allowed; neither are spaces at the beginning of a line. Characters with bit 7 set are ignored.

Numbers may always be expressed as decimal or hexadecimal, the latter being preceded by a "£" sign, in the range 0-65535 inclusive. Any number of leading zeroes is allowed. Signed numbers are only allowed (i) when defining the assembler offset and (ii) as part of expressions in operands in the source text.

3.2. The status line

This is the top line of the display and is used for workspace information and error messages. The complete format is:

```
  GENER-80   SCE aaaa   OBJ bbbb   xxx ERROR
```

SCE aaaa and OBJ bbbb are the source and object counters respectively (see Sections 6.1 and 11.1). On sign-on, the program version and serial number are displayed instead of the source counter.

4. INITIALIZATION

Gener-80 should initially be entered at 1000 hex. This is the "cold start" and should always be used if the workspace areas have not been initialized as below.

Gener-80 first checks itself for data errors caused by loading problems or user program malfunction etc - a SYS ERROR message will be displayed and control will return to Nas-Sys if any are found, in which case Gener-80 should be reloaded.

The last line of the sign-on message is "SCE MEM?" which is a prompt for source workspace.

Gener-80 requires the user to define both the source and the object workspace before use. These may not overlap Gener-80 or its workspace or each other, and an error message will be obtained if this is attempted.

These workspaces may be placed anywhere in free memory and, once defined, Gener-80 will not write outside them. Therefore other programs which may be resident in memory are protected.

When allocating source workspace, you should allow extra space for the symbol table which will be appended to the end of the source program if an assembly is requested - this requirement depends on the amount of labels used but is typically 15% of the length of the source program. The source workspace should in any case be at least 256 bytes long if you are going to load a program from tape. Our advice is to be generous with the allocations so as to avoid any need to redefine them later.

When the SCE MEM? prompt is displayed you can define the source workspace by two numbers separated by a single space, eg £D00 £DFF would define a workspace from 0D00 to 0DFF hex inclusive. A SCE MEM ERROR message will be displayed if invalid, excess or insufficient numbers are entered, if the first number is equal to or greater than the second, or if the workspace so defined overlaps Gener-80 or its workspace.

The OBJ MEM? prompt is then displayed and the object workspace should be defined, again by two numbers separated by a single space. An OBJ MEM ERROR message will be displayed if invalid, excess or insufficient numbers are entered, if the first number is equal to or greater than the second, or if the workspace so defined overlaps Gener-80 or its workspace or the source workspace earlier defined.

Finally, the OFFSET? prompt is displayed. The assembler offset (see Section 11.4) should be specified by a single number, which may be preceded by "+" or "-". The use of the ESC key will give a zero entry. A SYN ERROR message will be displayed if an invalid number is entered.

Once initialization is complete, the source workspace is cleared and the main command loop is entered.

There is a "warm start" at 1003 hex which does not clear the source workspace, but this should only be used if Gener-80 has been initialized first as above. Like the cold start, it checks Gener-80 for data errors, but in addition it compares a checksum of the source workspace with one stored when Gener-80 was last exited - a SCE MEM ERROR message will be displayed if they do not match, in which case the source program should be reloaded from tape.

5. THE MAIN COMMAND LOOP

5.1. Introduction

The main command loop is entered from both cold and warm starts, and resets the stack pointer to avoid any possible problems with user stack overflow. The prompt line is:

```
  COMMAND?  A  D  E  L  L?  M  O  P  PA  PAT  S
```

cursor is positioned on the next line, ready to accept one of these commands, which have the following meanings:

```
  A - assemble the source program
  D - delete the source program
  E - edit the source program
  L - load a source program from tape
  L? - verify a source program on tape
  M - move a block of source program
  O - object workspace and assembler offset redefinition
  P - print source program
  PA - print an assembled source program
  PAT - print an assembled source program with sorted symbol table
  S - save source program to tape
```

The D, M, P, PA and S commands can use optional labels to specify which part of the source program to perform the action on. The E command can be used to find a particular string in a label, operand expression or comment field. For further details, consult the sections dealing with these commands.

The main command line is always returned to when any of the above commands has been completed. Normally the prompt line appears at the top of the main display area, but in the L, L? and S commands it follows the tape block listing so any errors in the last few blocks can easily be seen. Escape back to Nas-Sys is accomplished by pressing the ESC key; a checksum of the source workspace is stored when this is done (see Section 4).

5.2. Error handling

The error messages encountered when entering a command line are:

CMD ERROR - either the first letter was not a valid command or else an unassigned control key was pressed

LAB REF ERROR - a label was specified which could not be found or there was an illegal use of two labels (eg label1 occurs after label2)

STR LEN ERROR - more than 14 characters used in a find string (see Section 6.5)

SYN ERROR - illegal command letters, misuse of spaces or wrong number of labels specified

6. CREATING AND EDITING A PROGRAM

6.1. Introduction

The command to enter the editor is:

```
  E string
```

where the string is optional. If specified, the cursor will be positioned at the start of the first line containing that string as part of a label, operand expression or comment (see Section 6.5 for further details). If no string is specified the cursor is positioned at the start of the source text and, if this is empty, the screen will be blank.

The source counter on the status line gives the address (in hexadecimal) of the next free byte after the end of source. This is updated as you add or delete lines so that you can always see just how large the text is.

6.2. Line format

Each line must consist of an assembler directive, a Z80 instruction mnemonic or a comment. All lines must start on the first column - no leading spaces are allowed. The formats are:

```
  LABEL: DIRECTIVE ;COMMENT
  LABEL: INSTRUCTION ;COMMENT
  ;COMMENT
```

Line numbers are not used. Appendix 1 gives a brief comparison of the conventions used by Gener-80 and the Zilog Z80 assembler.
Labels are optional except in EQU assembler directives. They may consist of up to six letters or numbers. The first character must be a letter, and the label must end in a colon, followed by one space.
Directives may be any of the following, where "exp" denotes an expression (see below). The action on assembly is also given:

DEFB exp enters exp (which is a single byte) into memory at the current location counter address. Multiple definitions are possible using a comma between each expression, eg DEFB 0,1,3
DEFW exp enters exp (which is two bytes long) into memory at the current location counter address. The most significant byte is entered last. Multiple definitions are possible.
DEFS exp allocates the number of bytes given by exp as a storage area, commencing at the current location counter address.
DEFM "m" enters the ASCII character string denoted by m into memory, commencing at the current location counter address. Note that the string should be enclosed in double quotation marks.
END terminates the assembly.
EQU exp equates the label field of this line with exp which may be up to 16 bits long.
ORG exp changes the location counter to the 16-bit value given by exp.

Labels used in DEFS, ORG and EQU expressions must be defined in preceding lines. ORG and DEFS directives must not decrease the location counter, eg DEFS -1 would cause an error message upon assembly.
Instructions may be any of the standard Zilog Z80 assembly-language mnemonics. A single space must separate the mnemonic and any operands. If two operands are present, they must be separated by a comma only, eg

```
  DEC HL
  EX (SP),HL
  LD A,(IX+0)
```

In those instructions specifying a bit, interrupt mode or restart, eg

```
  SET 7,A
```

```
  IM 2
  RST 0
```

the bit, restart address or mode number is treated as an expression (see below) so it is possible to use a label instead, eg

```
  RST ROUT
  BIT FLAG,H
```

Indexed address displacements are also treated as expressions, so

```
  LD A,(IX-BASE+DISP)
```

is acceptable. Because expressions are only evaluated upon assembly, it is possible to enter instructions such as BIT 8,A, although the range error would be detected if assembly was attempted.
Instructions using relative jumps should specify the destination address, eg JR START. The byte following an RST RCAL can be defined using DEFB address-$-1, although care should be taken if this is done to ensure its value is not outside the range -128 to +127.
Expressions which form part of an assembler directive or instruction operand field may be combinations of the following:

decimal number
hexadecimal number (prefixed by "£")
initial value of the location counter for that line (denoted by "$")
label symbol
single ASCII character (enclosed in double quotation marks)

These five types may be combined using "+" or "-" signs to form the complete expression. A leading sign is allowed. Examples of valid expressions are:

```
  PRGBOT-PRGTOP-£100
  "Z"+1
  $+OFFSET
```

Expressions are evaluated from left to right upon assembly using 16-bit integer arithmetic, eg -1 would be evaluated as FFFF hex.
Comments are optional. They must either start on the first column of the line or be separated from the rest of the line by a space. They must begin with a semicolon. An example line is:

```
  INC (IY+8) ;Increment page counter
```

Comments may consist of any character it is possible to enter. They are ignored by the assembler.

6.3. Cursor controls

The complete list of cursor and other controls available in the edit mode is given below:

Function	**Key**
Backspace	BACKSPACE
Cursor left	Left arrow or CTRL/Q
Cursor right	Right arrow or CTRL/R

Cursor up	Up arrow or CTRL/S
Cursor down	Down arrow or CTRL/T
Page back (15 lines)	CTRL/B
Page forward (15 lines)	CTRL/F
Cursor to start of source	CH or CTRL/W
Cursor to end of source	LF or CTRL/J
Cursor to end of line and enter	CTRL/E
Enter line	ENTER
Delete character	SHIFT/Left arrow or CTRL/U
Insert character	SHIFT/Right arrow or CTRL/V
Delete line	CTRL/D
Insert line	CTRL/I
Continuation	CS or SHIFT/BACKSPACE
Escape from editor	ESC or SHIFT/ENTER

If ENTER (or CTRL/I) is used, the editor will ignore all characters at or after the cursor. The purpose of the CTRL/E command is to avoid the effort of moving the cursor from, say, half way along a line of characters to the end by using CTRL/R repeatedly and then pressing ENTER. (No equivalent command exists for CTRL/I.)
The "page forward" and "page forward" commands can be used to step quickly through a program in either direction, 15 lines at a time (from the first line displayed, not the cursor position).
If it is wished to abort a partly-typed line, "cursor up" or "cursor down" will perform this function.

6.4. Error handling

When the line is entered it is checked for syntax and label definition errors. If any are found, one of the error messages listed below will be displayed and keyboard input will be inhibited until the "continuation" key is pressed. This prevents faulty lines being entered and ignored by the editor if the user is touch-typing and not watching the screen. Upon continuation, the faulty line will be cleared and the old one, if any, displayed.

CMD ERROR - an unassigned or illegal control key was pressed.
LAB DUP ERROR - the label has already been used elsewhere in the program.
LAB REF ERROR - an EQU assembler directive did not define a label.
LAB TYP ERROR - the label is an operand reserved word, ie a register name or flag status.
NUM ERROR - a number in an expression is greater than 16 bits long or has an illegal character, eg a letter in a decimal number.
STR CHR ERROR - the first character of a label (in the label or operand fields) is not a letter or else another character is not a letter or number.
STR LEN ERROR - a label used in the label or the operand fields has exceeded the maximum of six characters.
SYN ERROR - general syntax error from several possible causes such as instruction/directive not being recognized, incorrect combination of operands, misuse of spaces or commas etc.

A SCE MEM ERROR message will be generated if you attempt to enter a line which would overflow the source workspace. In this case control returns to the main command loop.

6.5. Finding a label or operand string

As mentioned at the start of this section, a facility exists for finding a string in a label, operand expression or comment field. The facility cannot apply to complete instruction mnemonics or directives because they exist in the source file in a coded form.
The string may be up to a maximum of 14 characters.
When this facility is used, the cursor will be positioned at the start of the first line which contains the string, ready for editing, and pressing the "continuation" key will find the next occurrence of the string after the current cursor position. If the string cannot be found, a return to the main command loop will be made. Requesting a continuation when no string was specified will result in a CMD ERROR message.
Some examples will show how this facility can be used:

E : will find all lines defining labels
E ; will find all lines containing comments
E LABEL: will find the line defining LABEL
E LABEL will just find the first mention of LABEL, which could be the defining line or an operand expression such as in JR LABEL
E LAB will find LABEL as above but also LAB1, FNDLAB etc
E 2 will find such lines as BIT 2,A and LD A,2
E 2, will find such lines as BIT 2,(IY+0) but not BIT 2,A or LD A,2
E 2,+DISP will find BIT 2,(IY+DISP), SET 2,(IY+DISP) etc

Note that register names etc cannot be found as they will have been compiled to code.

7. DELETING A PROGRAM

The command to delete a program is:

D label1 label2

where both label1 and label2 are optional, ie

D deletes the whole program
D label1 deletes the program from label1 onwards
D label1 label2 deletes the program from label1 to label2

fter the deletion has been carried out, control returns to the main ommand loop. Note that the source counter is not affected by this peration.
If you accidentally delete an entire program using this command, it may be possible to recover it. The source counter should still show the end of the lost program - make a note of this. Return to Nas-Sys and inspect the start of the source file, using the "M" Command. The first byte will be set to FF hex. Set this byte to 02 and the following one to 00. If the next byte is not 0D, set it to B. Now set address 0F19 to the original source counter reading, low yte first as usual. (If you have lost this reading you will need to go through the source file until you find the first occurrence of FF Hex and use this address).
Now enter Gener-80 at 1003 hex and go into the editor. You should find that the first line is a NOP instruction, possibly followed by

a semicolon and some rubbish, but that the rest of the program is OK. Correct the first line and also check that the end of the program is still there.

8. MOVING A PROGRAM BLOCK

The command to move a block of the source program is:

```
  M label1 label2
```

where the labels define the block to be moved, label2 being optional, ie

  M label1 moves all the program from label1 onwards
  M label1 label2 moves the program block starting at label1 and ending at label2

You will find the use of M label1 particularly convenient when you wish to add a new routine somewhere in an existing source program. Instead of having to insert it line by line using the editor, you can enter it as a block at the end of the program and then move it using the above command.
  When the command line has been entered, the prompt "TO?" is displayed, and label3 can be entered. This defines the destination of the block to be moved. Obviously, label3 cannot be located within the block to be moved, and a LAB REF ERROR message will be displayed if this attempted, or if label3 cannot be found.
  This command works by first creating a space for the block at label3, then moving the block to this location, and finally deleting the block at its original location. Consequently there is a temporary expansion of the source text. If this would overflow the source workspace a SCE MEM ERROR message is displayed and the command abandoned. It may in this case be necessary to move smaller blocks at a time to prevent overflow.
  After the move has been accomplished, control returns to the main command loop.

9. SAVING A PROGRAM ON TAPE

The command to save a program is:

```
  S label1 label2
```

where label1 and label2 are optional (see Section 7). After the command has been entered, Gener-80 will prompt with "NAME?" This allows you to assign a name to the tape file. The name should be no longer than six characters (any characters are allowed). If you do not wish to assign a name, just press ENTER. If you wish to return to the main command loop, press ESC.
  Once the ENTER key is pressed, Gener-80 will start to save your program, unless the name you have assigned is longer than six characters, in which case a STR LEN ERROR message will eventually be displayed and the command abandoned.
  Gener-80 does not use the Nas-Sys tape routines and the information displayed on the screen is rather different. Programs are saved in blocks of 256 bytes, each block being assigned a block

er (decimal), commencing with 1, which is displayed as the block is being saved.
  Upon completion of the save routine, control will revert to the main command loop, the prompt for this being displayed after the block number list. The recording can be verified using the L? command (see Section 10). We recommend that two tapes (main and back-up) are used for any long program you are creating. After each editing session, save the program to the back-up tape which then becomes the main tape and vice versa. This ensures that only a few hours' work is wasted should anything happen to the master tape.

10. LOADING A PROGRAM FROM TAPE

When Gener-80 loads a program into your computer's memory it automatically appends it to any program already present. (There are no line numbers and the tape files are position-independent.) This is a useful feature as it means you can add general-purpose routines from a subroutine library to a program you are writing. However, it does mean that if you only want the program you are loading you will have to make sure that any existing program is deleted first.
  The load command is "L". Gener-80 then issues the NAME? prompt as with the save command - it will only load the program with the file name given, which means it can search through a tape containing other tape files until it finds the right one. All file names encountered are displayed so you can monitor its progress through the tape. If you do not wish to specify a name, just press ENTER when the NAME? prompt is displayed. If you wish to return to the main command loop, press ESC.
  As each block is loaded into memory, the following information is displayed:

```
  N...  n  aaaa
```

where N... is the file name (if specified), n is the block number and aaaa is the address of the start of the block. The verify command L? works in exactly the same way except that data is not loaded into memory and aaaa is not displayed.
  You should ensure that the first block loaded is in fact number 1, and that the block numbers run consecutively. If a checksum error is found a "?" will be displayed, otherwise a "." will be shown. Any blocks marked "?" should be reloaded by rewinding the tape and trying again. Avoid pressing the keyboard while loading as this will introduce errors.
  It is most important that care is taken to avoid corrupted source files because only the editor performs syntax checking, not the assembler. If Gener-80 is used with a corrupted source program it may produce unpredictable results. Note also that label duplication is only checked when lines are keyed in - you should ensure that labels in appended programs do not duplicate those in the existing one. If necessary, you can check this by use of the CTRL/E control on each line defining a label in the appended program, which effectively re-enters it.
  If the program you are loading would overflow the source workspace a SCE MEM ERROR message will be displayed and control will return to the main command loop (although the error message you are actually likely to see after you stop the tape is CMD ERROR, due to the spurious input from the tape).
  After loading is complete, the source counter is updated and

control returns to the main command loop, the prompt for this being displayed after the last block number.

11. ASSEMBLING A PROGRAM

11.1. Introduction

The command to assemble a program is "A". Unless your program has specified the origin of the object program using an ORG directive, it will be assembled to commence at the start of the object workspace.
  Assembly takes place in two passes. In the first pass Gener-80 constructs a symbol table (appended to the end of the source program) containing all the labels you have used, together with their values. After this has been done, the source counter will indicate the next free byte after the end of the symbol table, and the object counter will indicate the next free byte after the end of the object program.
  In the second pass Gener-80 uses this symbol table to evaluate all expressions used, and writes the resulting Z80 machine code to memory.

11.2. Symbol table look-up

After assembling the program, Gener-80 will issue the NAME? prompt which allows you to specify any label and see its value. This is very useful for setting breakpoints etc when debugging. Any number of labels may be inspected - a LAB REF ERROR message will be displayed if the label cannot be found. Just press ENTER when you wish to return to the main command loop.

11.3. Error handling

Error messages encountered during assembly are:

  END ERROR - the program did not terminate with the END assembler directive.
  LAB REF ERROR - a label which had not been defined was quoted in an expression (in the case of a DEFS, EQU or ORG directive this would mean the label had not been *previously* defined.
  RNG ERROR - an expression gave a value outside its permitted range. This would occur if:
  (a) the bit number in a BIT, RES or SET instruction was not 0-7 inclusive.
  (b) the mode in an IM instruction was not 0-2 inclusive.
  (c) the address of a RST instruction was not 00, 08, 10, 18, 20, 28, 30 or 38 hex.
  (d) a displacement for an index register instruction was outside the range -128 to +127.
  (e) a displacement for a relative jump instruction was outside the range -126 to +129. (Does not apply in the case of the byte following an RST RCAL.)
  (f) an 8-bit expression had a 16-bit evaluation outside the range FF00 to 00FF hex, eg LD A,256

All the above errors result in a return to the editor with the

faulty line displayed ready for correction. In addition there are three possible workspace errors:

  OBJ MEM ERROR - the object code would have overflowed the object workspace (control returns to the main command loop and the object counter is updated to show the extent of the error)
  OBJ MEM ERROR - an attempt was made by a DEFS or ORG directive either to write outside the object workspace or to decrease the location counter (control returns to the editor, with the faulty line displayed)
  SCE MEM ERROR - the symbol table would overflow the source workspace (control returns to the main command loop).

The "O" command may be used to redefine the object workspace if necessary.

11.4. Use of assembler offset

Gener-80 can assemble object code at an address which is different to the logical address implied by the source program, the difference being the *assembler offset*.
  For example, suppose you wish to assemble an object program at 1000 hex. This cannot be done directly, because Gener-80 occupies this space. The procedure is therefore to assemble it at a free memory location, say, 8000 hex, with an assembler offset of zero, and debug in the normal way. The ORG assembler directives are then changed so that the program starts at 1000 hex, and the program is assembled again but this time with an assembler offset of +7000 hex (the "O" command may be used to redefine the offset). The object program is still assembled at 8000 hex but its logical address is now 1000 hex and, once copied to that location, will be executable in the normal way.

11.5. Debugging a program

The recommended procedure after entering a source program is as follows:

1. Request an assembly.
2. If an error message is obtained, correct the problem.
3. Repeat steps 1 and 2 until the source program can be satisfactorily assembled.
4. Make a note of suitable breakpoint addresses etc using the symbol table lookup facility or produce a listing using the "PA" or "PAT" commands.
5. Save the source program to tape.
6. Execute the object program via Nas-Sys.
7. If it "crashes", the source program and/or Gener-80 may have been corrupted. Reload the former if a SCE MEM ERROR message is obtained upon re-entering Gener-80.
8. Correct the source program, noting the modifications down (preferably on the printed listing) and re-assemble.
9. Repeat steps 6, 7 and 8 until the object program is debugged, then save it to tape.
10. Return to Gener-80 and save the source program to tape.

12. PRINTING A PROGRAM LISTING

Gener-80 offers three options: printing of a source listing, printing of an assembled source listing, or printing of an assembled source listing with sorted symbol table. Printing may be temporarily interrupted by pressing any key except ESC (which returns control to the main command loop) and resumed by pressing the "continuation key" (CS or SHIFT/BACKSPACE). *After printing has finished, press any key to return to the main command loop.*

12.1. Printing a source listing

The command to do this is:

```
  P label1 label2
```

where both label1 and label2 are optional (see Section 7).
  Once this command has been entered, Gener-80 issues the NAME? prompt which allows you to specify the running title at the top of each listing page. This may, for example, consist of the name of your program, the version and the date it was printed. The length of this title can be a full line of 47 characters. If no title is required, just press ENTER only. If return to the main command loop is required, then press ESC.
  After the title has been entered, Gener-80 will start to print your program listing, giving each page your title followed by a page number. Each line of the listing has the following format:

```
  n  L...  I...  O...  ;C...
```

where n is the line number (commencing at 1), L... is the optional label, I... is the assembler directive or Z80 instruction mnemonic, O... is its operand(s) and C... is the optional comment. These five columns are tabulated. Note that, since Gener-80 does not use line numbers, n is a printing line number included for reference purposes only - it might be of value if, for example, you write an article describing your program.

12.2. Printing an assembled program listing

The command for this is:

```
  PA label1 label2
```

and works in exactly the same way as the P command, except that the listing has two extra columns, the format being:

```
  aaaa  hh...  n  L...  I...  O...  ;C...
```

where aaaa is the contents of the assembler's location counter prior to assembly of that source line, and hh... are the first four bytes of the resulting Z80 machine code which was written into memory at that address. Both these columns are expressed in hexadecimal notation.
  Before printing, the entire program is assembled to check for errors - if any are found the error handling is the same as the assembly command (see Section 11).

12.3. Printing an assembled source listing with sorted symbol table

The command for this is "PAT", and works in exactly the same way as the PA command except the whole program must be printed and a sorted symbol table is appended. Each line of the symbol table has the following format:

```
  label1 aaaa   label2 bbbb   label3 cccc
```

where aaaa, bbbb and cccc are the respective values of label1, label2 and label3 expressed in hexadecimal.

APPENDIX 1. COMPARISON WITH ZILOG Z80 ASSEMBLER CONVENTIONS

Gener-80 has the following differences:

1. A colon is required after the label in the EQU assembler directive.
2. Only numbers and capital letters are allowed in labels.
3. The DEFL, COND, ENDC, MACRO and ENDM assembler directives are not supported.
4. Hexadecimal numbers should begin with "£". The use of the "H" suffix is not supported.
5. Binary and octal numbers are not supported.
6. Double quotation marks are used to enclose ASCII characters in expressions and in the DEFM directive.
7. The only operators allowed in expressions are "+" and "-".
Expressions are evaluated from left to right.

APPENDIX 2. PRINT ROUTINE CUSTOMIZATION

Gener-80 supports a printer by using the Nas-Sys "U" command to activate a serial output routine used by RST ROUT. The address of this routine is written into address 0C78 hex. When printing has ended, the "N" command is used to turn it off (see Nas-Sys manual).
  When Gener-80 is cold-started, it writes a jump instruction to address 0F00 hex. This jump is to the routine within Gener-80 which uses the "U" command, and a call is therefore made to 0F00 hex each time a print is requested. You can change this instruction to jump to your own activation routine, which could also initialize the PIO if necessary, for example.
  Your activation routine should preserve the IY register and end with a RET instruction, while your printer output routine must preserve all registers and also end with a RET.