Volume 2: PROGRAMMING INFORMATION

Part 6: APPLIED PROGRAMMING

Section 4: SHELLSORT

CONTENTS

Chapter 1: INTRODUCTION

1.1 Purpose

To sort a number of fixed length records (data items) held in core store into ascending or descending order.

1.2 Summary

A record must be a number of consecutive words. The file to be sorted must consist of consecutive records in core store. Records are split into two parts, the data-area and the key (the part on which the sorting is to be performed). The key itself is split into parts which are to be sorted on individually. Any part of the key may occupy between 1 and 18 consecutive bits in any one word of the record. Information about the file and the way in which it is to be sorted is given in a sort-table and a sort-list. The former specifies details of the file and the latter details on how the file is to be sorted.

1.3 Form of Distribution

Shellsort is distributed as a mnemonic tape for input under 900 SIR.

1.4 Method of Use

Shellsort is assembled as a block of the users program and entered in the standard manner, storing the link in SHELLSORT and transferring control to SHELLSORT+1. A parameter word immediately following the entry must contain the address of the sort-table. Exit is to the location following the parameter word.

Example

```
11 SHELLSORT
8  SHELLSORT+1
0  SORTTABLE
```

1.4.1 Entry and Exit Conditions

The content of the A, Q and B registers is irrelevent on entry and undefined on exit. On exit the file will have been sorted

'in situ' by exchanging complete records.

1.5 Restrictions

The following restrictions on the use of SHELLSORT should be noted:-

(i) The file must be of the specified format.

(ii) If a file is sorted 'n' deep, then the two arrays SHIFTS and COLLS in SHELLSORT must be declared as >+(n+1). On the standard tape SHIFTS and COLLS are declared as >+1Ø, i.e. sorted 9 deep is permitted.

(iii) All records must have identical format.

Chapter 2: FUNCTIONS

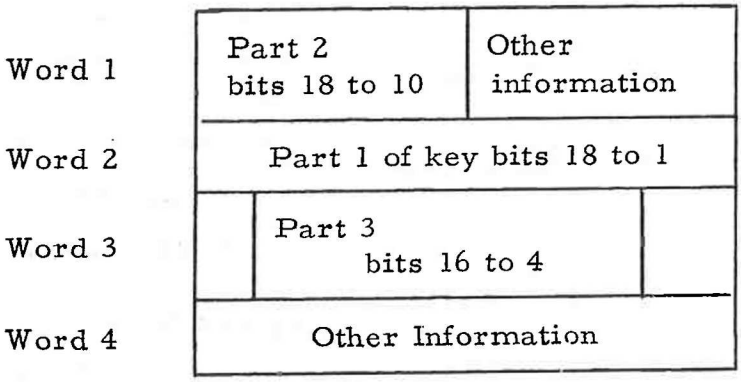
2.1 Format of File

The records to be sorted may be of one or more 18 bit words. They must reside in a continuous area of core store and must all be of the same length. The values on which the sort is to depend are held in a key, which must be of the same format in every record.

The key may form part or all of the record, it may consist of one or many parts, (the maximum number of parts is 9 in the standard program, but see Chapter 1.5).

Each part of the key may occupy the whole or part of an 18 bit word in the record. One part of a key cannot extend over more than one 18 bit word in core store, but see Chapter 2.5.

Example: 4 word Record, 3 part key



2.2 Sort-table

The sort-table is of the following form:-

```
SORTTABLE    +400          (number of records)
             +3            (number of words per record)
             Ø  FILE       (the address of the first location
                            of the file)
             Ø  SORTLIST   (the address of the sort-list)
```

+2	(the word of the record containing part 1 of the KEY (The first word of the record is word 1)
&ØØØ777	(The collating constant for part 1 of the KEY i. e. in this case part 1 of the KEY is the 9 least significant bits of word 2)
+3 &ØØ1777)	as above but for part 2 of KEY.
etc.	
+Ø	(end of sort-table).

2.3 Sorting on Negative parts of the Key.

Any part of the key consisting of n consecutive bits may be regarded as

i) An unsigned n bit number in the range 0 to $2^{(n-1)}$ i. e. always positive.

OR ii) An (n) bit number with bit (n) representing the sign bit, i. e. negative values are permitted, (stored in twos complement form, as for 18 bit machine words)

To specify the way in which any particular part of the key is to be interpreted (i. e. as above), the part numbers in the sorting priority table of the sort-list should contain +n (where 'n' is the relevant part of the key) if this is to be interpreted as (i) and -n if it is to be interpreted as (ii). The absolute value of the number entered will be taken as the relevant part of the key.

2.4 The sort-list

The sort-list is of the following form:-

SORTLIST	+1	(sort file in ascending order, would be -1)
		(if file to be sorted in descending order)
	+5	
	+3	
	+4	
	+6	(see below *)
	+1	(and 2. 3)
	+2	
	+Ø	

*This table specified the priorities in which the parts of the KEY are to be sorted.

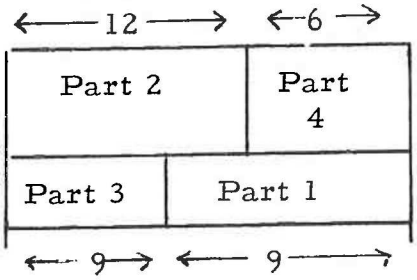
e. g. In this case the file is sorted on part 5 of the KEY. If two entries are found equal in part 5, then these two records are sorted on part 3 etc.

The word containing each part of the KEY and its collating constant are given in the SORT-TABLE.

The sort-table and the sort-list must be supplied by the user. The address of the sort-table must be placed in the location following the entry to SHELLSORT.

N. B. The order in which the parts of a KEY are numbered is arbitary. Having numbered the parts, however, the items in SORTTABLE +4 onwards must be set-up accordingly.

e. g. A KEY of the form

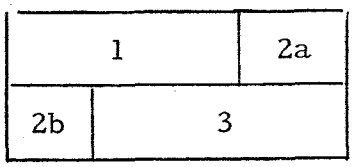


must be specified as

```
+2
&ØØØ777
+1
&7777ØØ
+2
&777ØØØ
+1
&ØØØØ77
+Ø
```

2.5 Sorting 'end-around'

It is possible that the key of a record may contain a part stored 'end-around' i. e.



Parts of PART 2 of the KEY are stored in both words 1 and 2 of the record. This may be sorted on by specifying the sort on part 2a, and, if these are found equal, then sort out part 2b. The parts must be numbered 1, 2, 3, 4 for 1, 2a, 2b, 3 respectively. This will achieve the same effect as sorting on part 2 stored as consecutive bits. Only positive parts of keys may be sorted on in this manner.

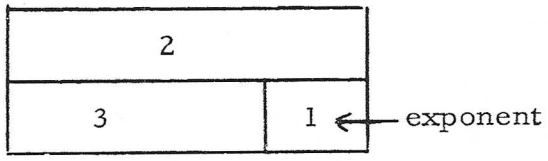
2.6 Character Sorts

Internal and/or tape code characters may be sorted using SHELLSORT by representing each character as a separate part of the key.

2.7 Sorting Floating Point Numbers

2.7.1. Packed Numbers

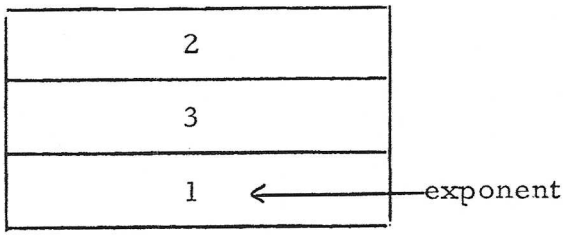
Packed numbers are regarded as two word records. Sorting is on firstly, the exponent, secondly the most significant bits of the mantissa and lastly on the least significant bits of the mantissa. i.e. the sorting sequence is:-



Negative numbers must be permitted in parts 1 and 2.

2.7.2 Unpacked Numbers

Specify 3 whole word keys, on the order : exponent, most significant mantissa, least significant mantissa.



2.8 Use of Extra Modules of Core Store

The main program, the sort-table, the sort-list and the file may be stored in different modules, providing that when the address of (a) the sort-table (b) the sort-list and (c) the file, is specified in (a) the parameter word (b) the sort-table and (c) the sort-table, the value specified is the address relative to the module containing SHELLSORT. e.g. If SHELLSORT is in module 1 and the file in module 2, then the sort-table should specify 1 FILE (i.e. 8192 + module 2 address of FILE), as the address of the file, regardless of the module containing the sort-table. Shellsort must be in the same module as the main program. The file may extend over

more than one module of core store, without restriction, except that it may only overwrite locations 8180 to 8191 of module 0 if the initial instructions are disabled.

2.9 Program Levels

Shellsort may be run at any program level.

Chapter 3: METHOD USED

The method used is a high speed sift sort technique with a varying interval of comparison and exchange. The method is described in "A HIGH SPEED SORTING PROCEDURE" by D. L. SHELL in "Communications of the A. C. M." Vol. 2 No. 7 of July 1959.

Chapter 4: SPEED AND STORE USED

4.1 Examples of Time Taken

On the 903, the time taken to sort 1000 seven word records, sorting 9 deep (i.e. each key consists of 9 parts) was approximately 2.5 minutes.40 three word records, sorted 6 deep, took approximately 5 seconds.

Equivalent times on 905, with one microsecond store, are 20 seconds and 0.6 seconds respectively.

It must of course be emphasised that times depend largely on the random nature or otherwise of the records, especially the number of records that can be sorted on the first part of the key.

4.2 Store Used

The store used is approximately 220 locations of code, and 17+2n (where n is the sort depth) locations of data; plus, of course, the users file, sort-table, and sort-list.