(51) International Patent Classification 4: G06F 5/00, H03M 7/30

(11) International Publication Number: WO 88/02144

(21) International Application Number: PCT/SE87/00406

(22) International Filing Date: 9 September 1987 (09.09.87)

(31) Priority Application Number: 905,237

(32) Priority Date: 9 September 1986 (09.09.86)

(33) Priority Country: US

(71) Applicant: INVENTRONIC DATA SYSTEMS AB [SE/SE]; Smedsättvägen 66, S-161 39 Bromma (SE).


(81) Designated States: AT (European patent), AU, BE (European patent), CH (European patent), DE (European patent), DK, FI, FR (European patent), GB (European patent), IT (European patent), JP, LU (European patent), NL (European patent), NO, SE (European patent).

(54) Title: ARRANGEMENT FOR DATA COMPRESSION

(57) Abstract

System (i) for putting in store an Entity Key (EK) set; (ii) for providing a response to a subsequent encoded input of a particular EK (INPUT SEQUENCE OF OCTET I/O CODE-WORDS REPRESENTING THE EK) - such response being an encoded output of a matching/non-matching condition, a pointer value and/or an EK Sequence number (EKS); and/or (iii) for providing a response to a subsequent encoded input of a particular EKS - such response being an output sequence of I/O code-words representing the corresponding particular EK. The system comprises (i) code conversion means (46) transforming the current EK input into a number of linked digital words (L1 through LZ), which thereby attain individual current link values and jointly a current overall link value representing the EK input in a compact form, compaction being performed using various means (e.g. BIT MAP MEMORY MEANS); (ii) a mandatory first link memory unit (12) holding previously stored EK related first link values (L1); (iii) a non-mandatory intermediate link memory unit (58) holding previously stored EK related intermediate link values (L2 and L3); (iv) a mandatory main link memory unit (14), holding an ordered set of previously stored main link values (L4), each member being related to at least one member of the EK set; and (v) a non-mandatory additional link memory unit (50) holding previously stored EK related additional link values, if any, (L5 through Lz). Use of various data compression techniques, particularly within input code conversion means, facilitates compact storage of large EK sets and contributes to very short response times in search operations, but also to fast sorting at input of additional, previously not stored, Entity Keys. Hence, searching and sorting speeds are reduced, e.g. in data base applications, EKS code-words may also be applied to represent data base file segments per se, e.g. words and/or phrases of a text file, for compact file storage and for fast file transfer.
FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

<table>
<thead>
<tr>
<th>AT</th>
<th>Austria</th>
<th>FR</th>
<th>France</th>
<th>ML</th>
<th>Mali</th>
</tr>
</thead>
<tbody>
<tr>
<td>AU</td>
<td>Australia</td>
<td>GA</td>
<td>Gabon</td>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
<td>GB</td>
<td>United Kingdom</td>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>HU</td>
<td>Hungary</td>
<td>NL</td>
<td>Netherlands</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
<td>IT</td>
<td>Italy</td>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
<td>JP</td>
<td>Japan</td>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
<td>SD</td>
<td>Sudan</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>KR</td>
<td>Republic of Korea</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td>LI</td>
<td>Liechtenstein</td>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>LK</td>
<td>Sri Lanka</td>
<td>SU</td>
<td>Soviet Union</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
<td>LU</td>
<td>Luxembourg</td>
<td>TD</td>
<td>Chad</td>
</tr>
<tr>
<td>DE</td>
<td>Germany, Federal Republic of</td>
<td>MC</td>
<td>Mosaco</td>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>MG</td>
<td>Madagascar</td>
<td>US</td>
<td>United States of America</td>
</tr>
</tbody>
</table>
ARRANGEMENT FOR DATA COMPRESSION

A. DESCRIPTION

1. Definitions of Terms as used herein
   (a) General  (b) User Environment  (c) System Environment

   5 (a) ATTRIBUTE: An inherent characteristic; ELEMENT: A distinct irreducible
       something; ENTITY: A distinct something possessing unique qualities because
       of its attributes; SET: A number of distinct 'somethings' that belong
       together; MEMBER: A distinct something that belongs to a set; ORDERED SET:
       A set having all its members arranged in a distinct sequential order;

   10 ITEM: A member of an ordered set; ORDINAL: An item sequence number; BIT OF
       INFORMATION: A unit of information equivalent to the result of a choice
       between two equally probable alternatives.

   (b) SYMBOL: A visual symbolic representation of something; CODE SYMBOL: A
       member of a symbol code; SYMBOL CODE: A set of graphic and/or control code
       symbols designed for the purpose of graphic recording of data; LETTER:
       Anyone of several graphic code symbols that combine to form graphic words,
       such as textual and lexical words; NUMERAL: Anyone of several graphic code
       symbols that combine to form numbers; BINARY NUMERAL: Anyone of the two
       numerals 0 and 1 that combine to form binary numbers; DECIMAL NUMERAL:

   20 Anyone of the ten numerals 0,1,2,3,4,5,6,7,8,9 that combine to form
       decimal numbers; HEXADECIMAL NUMERAL: Anyone of the sixteen numerals
       0,1,2,3,4,5,6,7,8,9,A,B,C,D,E and F that combine to form hexadecimal
       numbers - for clarity a radix indicator 'X' may be appended to numerals
       and numbers, e.g. FX, FFX; ALPH ADECIMAL NUMERAL: Anyone of the thirty-two
       numerals 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H,I,J,K,L,M,N,P,Q,R,S,T,U,V AND
       W that combine to form alphadecimal numbers, for clarity a radix indicator
       'α' may be appended to numerals and numbers, e.g. WX, WWα; SIGN: Any other
       graphic code symbol used with letters and/or numerals for the purpose of
       graphic data recording, e.g. punctuation mark; CONTROL CODE SYMBOL: A code

   30 symbol representing a control function, e.g. an instruction to a printer
       to execute line feed, new page etc.; SP or SPACE: A control code symbol
       representing the control function of generating a separation between
       recorded code symbols; NULL or NULL: A control code symbol representing a
       nullity; RECORD: A graphic recording of data based upon a symbol code;

   35 SEGMENT: A sequence of code symbols constituting a record of a text
       segment, a data program segment or a data aggregate segment, e.g. a
textual word or phrase of a running text; FILE: A series of segments related to each other and ordered in a distinct sequence; TEXT FILE: A file containing segments forming a running text; PROGRAM FILE: A file containing segments forming a data program; DATA FILE: A file containing segments forming a data aggregate; FREQUENT SEGMENT: A segment occurring frequently within a particular class of files; UNFREQUENT SEGMENT: A segment occurring unfrequently within a particular class of files; REFERENCE SEGMENT: A frequent segment selected as a reference; RSS: A reference segment sequence-number within a set of reference segments; ENTITY RECORD: A record about an entity; ENTITY RECORD SET: A set of entity records; EX (ENTITY KEY): an identifier of a concrete entity record within an entity record set, the entity key being identical to such entity record or a part of a larger such entity record that also contains further data; EKS: An entity key sequence-number, i.e. an EX ordinal within an EX set; EX ATTRIBUTE: An attribute distinguishing one or several members of a set of entity keys; EX ATTRIBUTE SET: A selected set of EX attributes, e.g. a set of textual word suffixes; I/O: Input to and/or output from a data system; I/O CODE: A set of relations between I/O code symbols and I/O code-words (see below), facilitating a user/system interface.

SYSTEM: Interacting arrangements including one or several means described herein; I/O CODE-WORD: A distinct system representation of an I/O code symbol in accordance with a specific I/O code; I/O CODE-WORD VALUE: A distinct numerical quantity denoted by symbol and name using e.g. binary, decimal or hexadecimal notation (Ex.: Using as I/O code the Extended American Standard Code for Information Interchange, the I/O code symbol '4', representing the decimal numeral named 'Four', would correspond to an I/O code-word with a binary value of '00110100', equivalent to a decimal value of '62' named 'Fiftytwo' and equivalent to a hexadecimal value '34' named 'Thirtyfour HEX'; STORE (the verb): To put in store - not used in the context of 'hold in store', which implies that the term 'previously stored value' means a value that is still being held in store; The term 'store' and the term 'put in store', as used herein, does not exclude the context of storing data within read only memory devices, e.g. in a mass production process; STORING: An act of putting in store; STORAGE: A state of being held in store, also a place for holding in store; ENTITY DATA SET: A system representation of an entity record set; EX SYSTEM: An entity key system, i.e. a system for holding an EX set in store, facilitating retrieval of related entity records by a user and/or retrieval of related entity key representations for internal use within the system; COMPRESSED
FILE: A compact file representation being held in store or transferred between separate locations; FILE HANDLING SYSTEM: A system for storage and transfer of file representations, e.g. storage and transfer of compressed files; DIGIT: a system representation of a numeral, the numerical value of which is being held within the system, either in a permanent read only memory or in an erasable read/write memory; BIT: A binary digit, i.e. a digit from a radix two number-system based on the two elements 0 and 1, hence a binary digit requires a two-state memory cell for storage; BIT OF STORAGE: A unit of storage capacity representing a capacity to hold one binary digit; BYTE (OF STORAGE): A unit of storage capacity representing a capacity to hold eight binary digits; BYTE LOCATION: A physical storage device having a storage capacity of one byte; TERNARY DIGIT: A radix three digit, requiring a three-state memory cell for storage; NIBBLE: A radix 16 digit, requiring four bits of storage; ALPHIT: An alphadecimal digit, i.e. a radix 32 digit, requiring five bits of storage (the name alphit was chosen because a set of 32 alphits is enough to represent a complete set of lower or upper case letters of most alphabets currently in use); OCTET: A radix 256 digit, requiring one byte of storage; DIGITAL WORD: A sequence of digits being held within a functionally contiguous string of memory cells or being transferred from one location to another, whenever being referred to or shown in symbolic form the leftmost digit position of the sequence or string is supposed to hold the most significant digit; LINKED DIGITAL WORD (CONCATENATED WORD): Two or more separate digital words having been functionally combined into one contiguous sequence; VALUE: A distinct numerical quantity uniquely representing a digital word, the term value being used to denote anything that is being held within a string of memory cells or being transferred from one location to another, such value being denoted by symbol and name using e.g. binary, decimal, hexadecimal or alphadecimal notation; OVERALL VALUE: A value of a linked digital word; CODE-WORD: A digital word representing a member of a code; CODE-WORD SET: A set of fixed length or variable length code-words representing all members of a code; ORDINAL CODE-WORD: A sequence-number representation of a member of an ordered set; CODE-WORD VALUE: A value denoting a code-word; COMPACTION CODE: A system code facilitating conversion between one code-word set and another more compact, i.e. less redundant, code-word set representing the same code members; COMPACT CODE: A low redundancy code created by compaction code conversion of a non compact code; COMPACT CODE-WORD: A digital word representing a member of a compact code; LINK: A storage device in a chain of functionally linked storage devices;
LINK VALUE: A value denoting the content of a link; ADDRESS WORD: A digital word representing an absolute or a relative address; ADDRESS VALUE: A value quantifying an absolute or a relative address; ADDRESS COMPONENT: A digital word used in combination with other address components for assembling an address word; ADDRESS COMPONENT VALUE: A value denoting an address component, i.e. a value used in combination with other address component values for computing an address value; TABLE: A read/write or a read only memory area functionally arranged in rows of one or more columns for storing values; BOUNDARY ADDRESS TABLE (FIRST LINK MEMORY UNIT): A table generating as an output one or several boundary addresses in response to an input of a first link value, the table also generating, if required, in response to input of a boundary address, an output of a relevant first link value as well as other boundary addresses related to such first link value, if any; REDUNDANCY: Unused capacity of a storage location or of a set of code-words, i.e. the number of possible values not being used; REDUNDANCY RATIO: The ratio of the number of values not being used to the number of values being used (Ex.: A byte of storage used to hold nothing but one nibble. If each one of sixteen possible nibble values are being used, the redundancy ratio is (256-16)/16=15).

Some terms defined above in user environment are sometimes used also in system environment in the sense of a corresponding system representation, e.g. RECORD, SEGMENT, FILE, ENTITY KEY, EX ATTRIBUTE, EXS and RSS.

2. Introduction and background

The present invention relates to processes and apparatus for storage and transfer of data in computer systems and more particularly to arrangements for storage and transfer of compressed data. Two related arrangements, for use in combination or separately, facilitate compact storage of entity key sets for fast retrieval of data related to individual entity key inputs.

Further, an entity key system may be used as a tool for compact storage as well as fast transfer of complete entity data sets. A file handling system may use any one or a combination of the two arrangements to provide non-redundant sets of entity key code-words to represent file segments, thereby enabling compact storage and fast transfer of files of various kinds.

Known methods for achieving a compact data system storage of an entity key set include front-end and rear-end compression techniques, explained by C. J. Date in the third edition of 'An Introduction to Database Systems', Addison-Wesley, page 51-52, and by James Martin in the second
edition of 'Computer Data-Base Organization', Prentice-Hall, page 517-526. Sophisticated word processing programs are using similar methods to access and compress a set of textual words, for use as a correct-spelling reference. Such programs typically use 150,000 bytes of storage to hold 50,000 different textual words and the time required to scan a text file representing an ordinary page of text amounts to several seconds.

3. Disclosure of Invention

A first type of entity key system according to the present invention produces an input response in form of an indication whether an individual entity key that is being input is identical to any member of a set of entity keys that is held in store. An example is the use of an entity key system as a correct-spelling reference, where the set of entity keys that is held in store is a set of frequently occurring textual words and the entity key input is an individual textual word from a running text that is being tested. Storage of 50,000 different textual words would require less than 150,000 bytes of storage and the time required to scan a text file would be substantially shorter than the time mentioned above referring to a typical word processing application.

A second type of entity key system according to the invention produces an input response in form of either (i) an EKS code-word, i.e. an Entity Key Sequence-number representation that uniquely identifies an individual entity key within an entity key set or (ii) a reject signal in case an invalid entity key has been input. This second type of entity key system can, for example, serve as a vocabulary reference, providing EKS representations of common textual words and phrases. The purpose with this use of the entity key system is to provide compact system representations of all or most segments of any file, each such compact system representation being an encoded EKS, for such use referred to as a Reference Segment Sequence-number code-word or RSS code-word. A substantial compression of text files can be achieved by substituting relevant RSS code-words for frequently occurring text segments, such as common words and phrases. The degree of compression may be enhanced by applying variable length code-words to represent more or less frequently occurring segments as well as more or less frequently occurring individual symbols of infrequently occurring segments. When regenerating original text, the code-words are decoded to form relevant text segments, i.e. textual words and phrases.

A third type of entity key system according to the invention produces an input response in form of either (i) a pointer address uniquely related to
the entity key being input or (ii) a reject signal in case an invalid entity key has been input. Any data associated with the entity key may be transferred to and/or output from an external device, the location within such a device being found by use of the pointer address. An example of use of this third type of entity key system would be as a tool to provide fast access to the entity records of a complete computer based encyclopedia. Each entry is assumed to be a lexical word, hence lexical words are being used as entity keys in this case.

Each one of the three system types mentioned above may also be operating in an entity key input state, facilitating storage of entity keys being input, thereby of necessity performing a sorting operation.

The efficiency of data compression arrangements constructed and operating according to the present invention may be enhanced by employment of conventional Huffman coding techniques. In 1952 David A. Huffman's paper 'A Method for the Construction of Minimum-Redundancy Codes' was published in Proc. IRE, 40(9). More recently the Huffman coding technique was explained by Gilbert Held in a Wiley Heyden Publication 'Data Compression', copyright 1983 and reprinted 1984.

4. Brief Descriptions of the Drawings

(a) FIG. 1A - 1F illustrate different basic features of entity key system embodiments using unsophisticated input means and only two linked digital words for interfacing input means with memory means.
- FIG. 1A represents a case where the EK input sequence is 'abc'.
- FIG. 1B shows how a previously not stored EK is being stored.

(b) FIG. 1C illustrates the reversed process of outputting an EK I/O code-word representation in response to an input of an EKS, i.e. a particular EK sequence number.
- FIG. 1D - 1F relate to EK systems having means for outputting pointer addresses in response to EK I/O code-word inputs. The embodiment shown in FIG. 1D is operating in an EK matching mode and the one shown in FIG. 1E & 1F in an EK non-matching mode.

(b) FIG. 2A - 2C illustrate three different coding schemes for code conversion from standard I/O codes to compact codes. FIG. 2A shows conversion to five-bit alphit code-words, FIG. 2B to four-bit nibble code-words and FIG. 2C to variable length code-words. These coding schemes are also applicable for retransforming compact code-word sequences into standard I/O code-word outputs.
(c) FIG. 3A - 3C relate to EX system embodiments with input means employing code conversion to 5-bit alphit code-words.

- FIG. 3A shows an embodiment holding 11 different EXs in store. The input sequence is 'etch', a 32 bit sequence compacted to a 20 bit compact code representation.

- FIG. 3B illustrates that the embodiment of FIG. 3A may be used to store the compounded word 'et cetera'. The figure also illustrates that an EX containing more symbols than can be represented still may produce a unique overall link value.

10 - FIG. 3C shows an embodiment based on bit map memory means. The input sequence is 'etcetera'. The bit map holds a set of marked bits, representing the same eleven EXs as shown in FIG. 3A.

(d) FIG. 4A & 4B illustrate means for facilitating a proper sorting order when capital letters and compounded words are included. Code conversion to alphit code-words is employed.

(e) FIG. 5A through 5C introduce the concept of additional links. Code conversion to alphit code-words is employed. FIG. 5B shows how a proper sorting order is maintained. FIG 5C illustrates the use of bit map memory means to remove redundancy within additional links.

20 (f) FIG. 6A - 6D introduce the concept of intermediate links. Code conversion to alphit code-words is employed.

- FIG. 6A explains the conventions applied for designating variables and values.

- FIG. 6B & 6C show embodiment variants, both using as input the sequence 'etaonrih'. The two variants use two different principles for storing address component values within the intermediate link unit.

- FIG. 6D illustrates how an EX system employing bit map memory means is used as part of code conversion means in an EX system embodiment otherwise employing ordinary memory means.

30 - FIG. 6E shows an embodiment coping with input sequences comprising up to fourteen symbols. Code conversion is employed in two steps.

(g) FIG. 7A - 7B relate to EX system embodiments with input means employing code conversion to 4-bit nibble code-words and variable length code-words, respectively.

35 (h) FIG. 8A relates to a file handling system embodiment employing variable length code-words.
5. Detailed Description

With reference to FIG. 1A through 7B, the functioning of preferred embodiments of an EK system according to the invention will be explained by following in detail how responses to various inputs are produced.

Specific EKs are assumed to have been input and stored beforehand as part of a large set of EKs. Such inputs may have been previously stored within read only memory devices in a mass production process or they may have been stored by inputting EKs into a working system. In the latter case the embodiment uses read/write type of memory devices and can be set in a mode for storing previously not stored EKs. Erasing stored EKs is an obvious reciprocal process, not further discussed herein.

As in most of the figures, FIG. 1A shows several strings of boxes, each one representing a string of storage locations. Such strings are drawn, either horizontally as within input means 10, or vertically as within memory units 12 & 14, always in an order of increasing address values from left to right or from top to bottom. Vertical strings 16 & 18 are placed underneath a horizontally drawn system bus 20. Address values locating particular boxes are increasing with increasing lengths of vertical lines, e.g. line 22 drawn between bus 20 and the top 24 of the box being located.

Such a vertical line is pointing downwards if the box location is directly addressed and upwards to indicate that the address of the box location is registered as the result of a search operation. A value being read from a box is indicated by a line 26 coming out of the box and pointing at the bus. A search for a particular value within a string is illustrated by a vertical line 28, designated with the search argument value L2. Line 28 is related to a series of boxes within a particular searching range, each such box being marked along the left hand side of the string. Each test being executed is marked by an ordinal, '0' denoting the box first being tested for a match between the search argument L2 and the value L2x held in store within that box. A star '*' is used to mark a box that need not be examined because a match has already been found. A match at test number 2 is indicated by a short horizontal line crossing the left hand side of the box holding the matching value L2x. In reality L2x values are being read out of the numbered boxes one by one and transferred along line 28 for comparison with value L2 within control processor block 30.

A system of indexing is used in this specification to designate, in an orderly manner, address values, data values as well as tables holding such values. Mainly four index variables are used, p, q, r and s. The q, r and s indices are only used in their respective positions within the
combinations pqr, pqr and pq. Index variables may be exchanged for actual values using alphanumeral numerals. The symbol Z represents the ordinal number of the last item within a set. All four indices are ordinal indices, each one having values within a series 0, 1, 2, ..., Z.

Index p is the item sequence number within the set of L1a values related to the 1.EK set representation being held in store. No L1a values are held in store as such, however, each L1a value is represented as a relative address of a storage location within string 16, each such location holding a related boundary address value B2a. The series of values from B2a to B2z determines the size of each particular table T2a. The B2a values have been assigned in ascending value order with increasing p values, with adequate steps to make each T2a table just large enough to hold all those individual main link L2ma values that have previously been presented for storage concurrently with a valid L1a value.

Index q is an item sequence number for each particular value of p, thus designating L2ma values individually within each table T2a. All L2ma values are held in one functionally consecutive string 18, within each table T2a in value ascending order with increasing address values. The series of composite pq numbers are thus forming a number series in ascending pq value order with ascending address values. Value L2ma is held at the lowest address B2a and value L2z at the highest address B2z−1. The B2z− value is used to determine the end of the last table T2a. A plus or minus sign completing an index number designates the next higher or lower number in the index number series. For example, if L2z− is held in store at the highest address within table T2a and the value of p is 1, then index p7+ is equivalent to pq=20.

The storage capacity of a box is shown by the width, i.e. 1/10 of an inch for each bit of storage capacity. Byte boxes are thus 8/10 of an inch wide. Broken frames are used to indicate that the storage capacity may be higher than the width shown.

FIG. 1A - 1F illustrate different basic features of system embodiments using unsophisticated input means and only two linked digital words for interfacing input means with memory means. Except for the reversed process shown in FIG 1C, input means are shown at the top of the figures in form of a block 10 including two short strings of storage locations, 32 & 34, string 32 for temporarily holding a representation of a current I/O code-word input and string 34 for temporarily holding two linked digital words.
String 32 provides storage for octet values $O_0$, $O_1$, and $O_2$, values directly obtained from three eight bit I/O code-words representing a three symbol EX input sequence. The values of the octets are directly transferred to the two linked digital words residing in the 16 plus 8 bit string 34. The 16 bit word attains the value $L1=O_0*256+O_1$ and the 8 bit word the value $L2=O_2$. Let us assume that the I/O code used for input is the Extended American Standard Code for Information Interchange (EASCII), further that previously stored EXs comprise all 512 possible three letter combinations of the first eight lower case letters of the English alphabet.

Link value $L1$ is used by the control processor 30 for producing address values to access the first link memory unit 12, from where boundary address values $B2_0$ are being read out. The link value $L2$ is used for comparison with values held in store within the main link memory unit 14, within an address range determined by boundary address value pairs, $B2_0$ & $B2_+$. The control processor 30 is communicating with all other parts of the system via the system bus 20. Processor 30 is executing a control program for each input sequence and governs output of a response via bus 20, whereupon input means 10 are being reset, making them ready to receive a new input sequence.

With the assumptions made there are $8^3=512$ different EXs held in store, the storage of which has generated $8^2=64$ different boundary address values $B2_0$ to $B2_z$, where $Z=63$. These 64 values are the addresses of the first storage location 36 of 64 distinct tables $T2_0$ to $T2_z$, where $Z=63$. Each such table is holding 8 values $L2_{z0}$ to $L2_{zp}$, where $p$ ranges from 0 to 63, a total of 512 $L2_{zp}$ values representing 512 stored EXs.

FIG. 1A represents a case where the current EX input sequence is ‘abc’. The decimal EASCII values representing the symbols ‘abc’ are 97, 98 and 99, respectively. The symbols ‘ab’ are represented as $L1=97*256+98=24930$. Only one symbol pair, ‘aa’, that has generated a lower $L1$ value, hence index $p$ assumes the value 0 for the pair ‘aa’ and the value 1 for the pair ‘ab’. There are eight $L2_{zp}$ values held in store in table $T2_z$, i.e. 97 through 104 in decimal notation, the third one, decimal 99, matching the current main link value $L2$. As shown in the figure a match was obtained as a result of a search starting at the first storage location marked 0 within table $T2_z$, a location pointed at by a boundary address value $B2_z$, obtained from a two-byte storage location within table $T1$ at an offset address governed by the current $L1$ value 24930. Said match was obtained in the eleventh byte storage location, counted from the lowest address $B2_0$, and an EXS value of 10 is therefore output. The EXS value is an item sequence
number within a series 0,1,2...511, numbers that uniquely represent the complement of 512 EXs having been previously input.

Within table T1 are also shown B2n values related to a group of L1n values from 25185 through 25188, L1n - L1n values that correspond to symbol pairs 'ba,bb,bc,bd'. Also shown is the highest L1n value 26728, corresponding to the symbol pair 'hh', and the last two B2n locations related to a not stored EX that would have generated an B2n value of 65535. All T1 locations relating to not stored EXs are holding the same B2n value as the one stored within a preceding T1 location. Index n is used in lieu of index p to emphasize that the value is not related to a valid EX. The designation L1n will be used later on as representing any possible L1 value, including unvalid ones. Apparently table T1 contains a lot of redundant information when used as in this example. When holding only 512 EXs in store, 257 byte of storage is required for each EX, provided a full length T1 table is used, having 65537 two byte rows. On the other hand, with such an embodiment very few computer instructions are required to obtain a response from the system.

The particular embodiment being described has a capacity to store all possible three-symbol combinations employing a 256 symbol code, i.e. the maximum possible in case of an octet code-word representation. The storing capacity would be 256³=16.8 million different EXs, requiring a three byte wide T1 table in lieu of the two byte wide table shown in FIG. 1A. Nevertheless, the amount of storage required by table T1 would be insignificant. In total 1.01 byte of storage would be required for each EX. On the other hand, the response time would increase as the search range within each T2n table would amount to 256 byte of storage. However, the response time could still be kept extremely short by searching such range using conventional binary search methods. A relevant notion is also that minimal response time requires all tables to be stored in fast access type of electronic memories, a requirement more easily met if a high degree of compression has been achieved.

FIG. 1B shows how a previously not stored EX is being stored. This figure represents the same embodiment as in FIG. 1A, but with the number of stored EXs having been reduced from 512 to 511. However, tables 16 & 18 will contain the same values as in FIG. 1A after storage of the EX sequence 'abb'.

Input of octet values representing 'abb', i.e. 97, 98 and 98, is generating a current L1 value of 24930. This value causes boundary addresses B21 and B22 to be read out of table T1. Processor 30 tests that
B2_0 is greater than B2_1 and initiates a search for the current L2 value 98 within the T2_1 table, specified by said boundary addresses. Value L2_30=97 is read out of the first storage location 36 and found to be less than the current value L2=98. Next value L2_1=99 is found to be higher than 98 and the search is halted as all remaining L2_m values are even higher, having been stored in ascending value order. An EX reject signal 40 is output via bus 20 and the system user may command control processor 30, if not already done, to react upon such reject signal 40 executing an insert instruction sequence. Such sequence includes (i) moving to the next higher address each previously stored L2_m value having a pq index higher than 10, (ii) inserting the current L2 value within the storage location of the previously stored value L2_11 and (iii) increasing all B2_m values by one unit for all p values greater than 1.

In case no EXs having 'ab' as their first two symbols had been previously stored, processor 30 would have found B2_2 to be equal to B2_1. No T2_1 table would exist and an EX reject signal 40 would be output based upon first link criteria. In such case the insert instruction sequence would include (i) moving to the next higher address each previously stored L2_m value having an address equal to B2_1 and higher, (ii) inserting the current L2 value within the storage location at address B2_1 and (iii) increasing all B2_m values by one unit for all p values greater than 1.

To avoid a time consuming move of large number of values spare storage locations may be arranged at convenient intervals.

FIG. 1C illustrates the reversed process of outputting an EX I/O code-word representation in response to an input of an EXS, i.e. a particular EX sequence number. Processor 30 adds the current input EXS value 10, received via system bus 20, to the boundary address value B2o and reads from string 18 at the resulting address B2o+10 a current main link value L2=99. In order to determine also the first link value L1 the T1 table 16 is searched for the highest B2_m value that is equal to or lower than such resulting address. A binary search is employed in order to reduce the number of tests required to find such B2_m value. After ten binary search tests, numbered 0 to 9 in the figure, a sequential search is started at B1o+24928, because at this point the target location is close and must be found at an address lower than the one tested as number 8. At three sequential tests, A, B & C, values lower than B2o+16 are read out of table T1 and at number D B2o+16 is found to be higher, a value that is found at address B1o+L1+1. Processor 30 is now able to determine the current first link value L1=24930 by subtracting B1o+1 from the address found.
Finally output means 42 transforms the current link values into a sequence of three I/O code-words. The first octet value $O_0$ is computed as the integer part of $24930/256 = 97$. The second octet value $O_1$ is computed as $24930 - O_0 \times 256 = 98$ and the third value $O_2$ is equal to the current main link value 99. These three values represent the symbol sequence 'abc'.

FIG. 1D - 1F relate to EX systems having means for outputting pointer addresses in response to EX I/O code-word inputs. The embodiment shown in FIG. 1D is operating in an EX matching mode and the one shown in FIG. 1E & 1F in an EX non-matching mode. The term EX matching mode is used to emphasize the fact that any output from the system, such as pointer address output and EKS output, relates to an EX input that is matching a previously stored EX. The term EX non-matching mode is used to emphasize the fact that system output relates to any EX input generating overall link values also falling inbetween overall link values representing previously stored EXs. This latter mode is therefore in the following being referred to as a gateway mode of operation.

The embodiments shown in FIG. 1D - 1F are similar to the one in FIG. 1A, except that a second column 44 holding pointer addresses $P_{2\alpha\beta}$ has been added to each $T_{2\alpha}$ table, each such $P_{2\alpha\beta}$ address value being related to a particular first column $L_{2\alpha\alpha}$ value. In the gateway mode, however, each $P_{2\alpha\beta}$ value is also related to any EX that, if being input, would cause $L_{2\alpha\alpha}$ values to be inserted inbetween such $L_{2\alpha\alpha}$ value and its closest neighbor $L_{2\alpha\alpha\beta}$.

Any data associated with the EX may be transferred to and/or output from an external device, the location within such a device being found by use of the pointer value $P_{2\alpha\beta}$. Such data are supposed to be strictly related to specific EXs in the matching mode of operation, in gateway mode the system will relate EX groups to common external device addresses.

It is also worth mentioning that at storage of previously not stored EXs, a new row is inserted within a relevant $T_{2\alpha}$ table and $L_{2\alpha\alpha}/B_{2\alpha}$ value pairs being held in rows at higher addresses have to be moved. A firm relationship between each main link value $L_{2\alpha\alpha}$ and its related pointer address value $P_{2\alpha}$ is thus maintained. However, memory space limitations may dictate that such second column 44 of table $T_{2\alpha}$ is placed within a low cost external storage device, in which case the EKS code-word may be used as a pointer into such an external table.

For convenience a different method to present code-word values has been introduced in FIG 1D - 1F. Whenever input symbols appear inside a box or within brackets they should be interpreted as denoting the corresponding
code-word value.

FIG. 1D illustrates how a pointer value P_{12} is found within column 44 of Table T_{21} and read out for output via system bus 20. Manipulation of data associated with the EK 'abc' is facilitated using the pointer value for addressing.

FIG. 1E illustrates an embodiment identical with the one shown in FIG. 1D except that a pointer output response is facilitated also for an EK input sequence that has not been previously stored. Therefore this embodiment is operating in the gateway mode. The sequence 'aai' is used as input and generates a first link value that has been previously input as part of stored sequences, therefore the gateway mode of operation does not impact the functioning of the first link memory unit in this case. Upon a no match search in table T_{20}, the current L2 value [i] is associated with a particular L2_{0m} value, being the highest possible but not higher than the value L2 itself. The figure uses the '>' sign to indicate that such condition is met at L2_{07} and a double headed arrow is indicating that a related pointer address P_{07} is thereby found. Also L2 values searching table T_{21}, being lower than L2_{0m}, are being related to P_{07}.

FIG. 1F shows an embodiment identical with the one shown in FIG. 1E except that the symbol sequence 'aia' is used as input. This input generates a first link value that has not been previously input as part of a stored sequences. A comparison being made between B_{2m} for L1 and for L1+1 shows no difference as value B_{2m} is being read out from string 16 for L1=24937 as well as for L1=24938. With the system operating in the gateway mode the control processor will then read out the pointer value P_{07} directly via system bus 20 from the second column 44 of T_{2m} tables at an address B_{2m}-1. The same would happen for any L1 value from 24937 through 25184. The next value, L1=25185, is a value generated by previously stored symbol sequences, i.e. those beginning with the pair 'ba'. In the figure a double headed arrow indicates the location of the pointer address P_{07}. A box with a pointer address P_{00} has been added to cover L1 values below 24929.

The embodiments described so far are capable of storing only three-symbol EKs, which is a severe restriction in most applications. Conversion to compact codes facilitates accommodation of an increased number of code-words within string 32, whereby an increased number of input symbols per EK is allowed. A compact code representation has the additional advantage of reducing memory space requirements, which provides for shorter response times as more EK representations may be simultaneously held within a given working memory area. The compact
representation leads to shorter response times also because fewer bits will become involved at each retrieval attempt. Furthermore, the use of compact codes may facilitate a shortening of string 32, which reduces first link memory requirements. A shortening by just one bit will cut in half the length of boundary address table 16.

FIG. 2A - 2C illustrate three different coding schemes for code conversion from standard I/O codes to compact codes, to be applied within input means 10. FIG. 2A shows conversion to five-bit alaphit code-words, FIG. 2B to four-bit nibble code-words and FIG. 2C to variable length code-words. These coding schemes are also applicable for retransforming compact code-word sequences into standard I/O code-word outputs.

The alaphit code of FIG. 2A is suitable for adaptation to sophisticated sorting order requirements. In FIG. 3A through 6E such code is used, illustrating various features, including sorting order features.

Nibble codes as well as codes using variable length code-words are unusable in applications having specific sorting requirements. However, such codes have other qualities that make them suitable where sorting order is of no concern. In FIG. 7A and 7B such codes are used, emphasizing the importance of a high degree of compaction.

The functioning of a system embodiment including compact code conversion is basically the same as explained with reference to FIG. 1A through 1F, once the link values are determined. The particular features illustrated in those figures are therefore applicable also to embodiments using compact codes.

FIG. 3A shows an embodiment with an input sequence of four octet I/O code-words representing the EX 'etch'. Code conversion is employed, compacting the 32 bit input to a 20 bit compact code representation based on 5-bit alaphit code-words. Code conversion means 46 are shown in the form of a look-up table 48, comprising a string of 256 alaphit code-words having its first alaphit location at address B0. Alaphit values A, are read from table 48 in sequence from n=0 to n=3 at addresses B0+n. The table below explains how the contents of the small string 32, wherein the succession of alaphit values is temporarily held, is being transferred to string 34, temporarily holding the two linked digital words, here represented by their link values L1 and L2. Alaphit values representing the code symbols are taken from the encoding scheme of FIG. 2A, shown in the table in alphadecimal notation as well as in the corresponding binary notation. The bit representation is shown as one contiguous succession placed within the middle row of the table. Link values are shown in hexadecimal notation.
The first link memory unit is block 12 in FIG. 3A. The T1 table inside comprises only 4097 rows in this case, as compared to 65537 rows in the embodiments of FIG. 1A through IF. Control processor 30 reads B2a boundary address values from T1 storage locations, produced from the link value L1.

For L1=4E0x B2a and B2e are read out, determining the search range within string 18, in which string eleven EKs are shown as previously stored. Respective L2ma values are shown in hexadecimal notation. EKs comprising less than four symbols have been allocated L2ma values under the assumption that each input sequence of octet I/O code-words has been filled up by adding code-words representing a nullity, i.e. NUL with an alphit value of 2. The main link value ECx, representing EK 'etcetera', is read out as the last item at the end of the searching range and found to match the current L2 value. As can be seen from the figure, those eleven previously stored EKs would be presented in proper sorting order if being output sequentially in EKS number order, using the technique of FIG. 1C. The long word 'etcetera' would be output as 'etc'.

FIG. 3B shows how an input of the previously not stored EK 'etcetera' would result in insertion of the current main link value L2=67x at an improper place, considering a desirable sorting order. A solution to this problem will be presented later with reference to FIG. 4B. For an explanation of the functioning at an insertion, reference is made to previous comments to FIG. 1B.

FIG. 3C shows an embodiment based on bit map memory means 52. It is presumed that the same eleven EKs as shown in FIG. 3A have been previously stored, one of them, i.e. 'etcetera', being used for current input. Input means 10, including code conversion means 46, are identical to corresponding means shown in FIG. 3A except for the small string 34 that has been increased in length. Section values, S1 through S8, are produced exactly as are link values. The new term section is introduced in lieu of the corresponding term link to facilitate description of EK systems wherein
the same small string may hold section values as well as link values, section values being related to bit map memory means and link values to ordinary memory means. An example of such combined use will be discussed with reference to FIG. 6D.

5 The embodiment shown in FIG. 3C has one section for each alphit, thereby making section values identical to alphit values, which facilitates use of just one set of storage locations to hold these values. In other words, the two short strings 32 & 34 may be replaced by one common string. A maximum of eight section values are accommodate for each EX, requiring a set of eight bit maps, i.e. tables T1 through T8. The sequences representing the eleven EXs vary in length from one to eight alphit values, however, in order to facilitate output of an EXS in response to an EX I/O code-word input, the code conversion means 46 adds to each such sequence an appropriate number of control function alphits to make them all eight alphits long. In order to achieve a proper sorting order such alphits has to be added after the alphits that represent the EX symbols. However, the embodiment shown in FIG. 3C illustrates a memory saving alternative, which is preferable in applications not demanding a proper sorting order. Control function alphits are added up front, whereby these alphits are represented only once within each bit map. Hence, the five symbols a, b, c, d and e are all represented within each one of tables T1 through T7 by just one bit marked as being in the 1 state. These marked bits are located at a relative bit address 02^m within each table, i.e. the third bit position counted from right to left within the first row of each table. This bit address value corresponds to the alphit value 2^m used to represent the system control function NUL.

Within table T8 a first group of 32 bit locations, at relative bit addresses 00^m-0W^m, is used to represent previously stored S8 values related to such previously stored section values S1 through S7 that are represented by the lowest address bit mark within respective tables T1 through T7. Therefore, in FIG. 3C the first 32 bit group within table T8 is used to represent single letter symbol inputs. The five marked bits within this group represent the stored symbols 'a,b,c,d,e', e.g. the leftmost bit position within the first row, at relative bit address 07^m, 35 represents the symbol having a code-word value of 7^m. This value represents the symbol 'c', as can be seen from the code table in FIG. 2A. It is interesting to note that any previously not stored single symbol EX may be added, just by marking the particular bit position within table T8 at a relative bit address equal to the symbol code-word value.
The current EX input 'etcetera' generates eight alphit values that are placed within the small string 32, whereby current section values S1 through S8 are determined. These values are used to produce current relative bit location addresses within respective tables T1-T8. According to FIG. 3C each bit map table comprises a string of byte storage locations, the relative bit address of the least significant bit within each byte being indicated along the right hand side of each table using alphadecimal notation. The address of any other bit location is determined from the figure by adding 0-7 when counting from right to left. The values used for this purpose in the embodiment being described are designated S1L-S8L, obtained by stripping off the three least significant bits from each of the values S1-S8. As can be seen from FIG. 2A the S1 value 9* equates 01001 in binary notation and S1L thus attains the value 001.

In table T1 the current relative byte address used for addressing is computed as S1/8*01. Shown in the figure is the corresponding relative bit address S1m=S1-S1L=08*. The control processor reads the value M1 out of this byte location and produces a bit mark match as value S1L is found to point at bit position 001 which has a bit value 1. The match means that an S1 value equal to the current S1 value has been previously stored.

For each of the succeeding tables, T2-T8, the current relative byte address is computed as S2/8*M1*4, S3/8*M2*4,...,S8/8*M7*4, in Fig. 3C shown as the corresponding relative bit addresses S2m through S8m, i.e. 1P*, 10*, 18*, 2P*, 38*, 4G* and 50* respectively. The values M1 through M7 are produced as a count, within each table T1-T7, of the number of marked bits at bit location addresses lower than the current one. A corresponding M8 value equates the current EXS value used as output.

In order to facilitate a fast count of marked bits a mark-count table 54, also named TM table, is employed as shown in FIG. 3C. This table is a 4-bit wide look-up table using as an relative entry address the overall value of a group of eight bits, i.e. a byte value. The number of marked bits within the byte is read out of table 54 in one fast operation. On the right hand side of table 54 entry relative address values V* are shown in hexadecimal notation.

In order to further speed up the counting process when many successive 8-bit groups have to be examined, an aggregate mark-count table 56 may be employed. Shown in FIG. 3C is such a table named table TM8, holding a series of aggregated mark-counts for groups of 32 bits. For example, the value M8* found at the relative byte storage address 5 represents the total marked bit count within the first five 32-bit groups of table T8.
This particular value, M₀=10, is read out as a first step to determine
the EKS value related to the current EK 'etcetera'.

The second and last step to determine the EKS value is to produce a
mark-count within the sixth 32-bit group of table T₈, wherein a marked bit
representing the last symbol 'a' has been found at relative bit address
55™. This marked bit is located within the first byte of the 32-bit group,
therefore it only remains to test if there are any marked bits within this
byte at bit addresses below 55™. The byte value Mₐ0 is read out and the
three leftmost bit positions are masked and the remaining overall value
which happens to become 00X is fed to the TM table 54 as a Vₒ entry value
and Mₒ=0 is read out. The EKS value is determined as EKS=M₀+Mₒ=10.

As the current EKS value is produced as a count of the number of marked
bits within the T₈ table at bit position addresses lower than the current
one, EKS values are generated in such order that all single symbol EK's
will appear, when presented in EKS order, as being sorted in proper
alphanumeric order. All two symbol EKs will next appear in sorted order,
thereafter all three symbol EKs and so on. This is because, within each
table succeeding the first one, groups of 32 bits have been allocated to
represent previously stored EKs in an order governed by the count of
marked bits within the immediately preceding table at bit location
addresses below the bit location related to such 32 bit group. The current
input EK 'etcetera', being the longest one within the set of eleven EKs,
will appear as no. 10 within the series of EKS numbers 0-10.

FIG. 4A & 4B refer to a modified embodiment, having means for
facilitating a proper sorting order as follows: E, e, ET, Et, et, ETA,
Eta, eta, ETC, Etc, etc, ETC., Etc., etc., et cetera, ETcETERA, Etcetera,
etcetera, ETCH, Etch, etc. The modified embodiment differs from the one
referred to in FIG. 3A & 3B by having an 8 bit wide TL0 look-up table 48
and a control program function that determines a Q value based on the
values of the leftmost 3 bits shown in table 48. FIG. 4A & 4B provides the
algorithm 49 to be used in order to make the overall link value such that
a proper sorting order is obtained. This is achieved by including the Q
value at the end of string 32, ignoring any single space or hyphen
code-word when creating the preceding aliph value sequence.

FIG. 5A through 5C refer to embodiments equivalent to the one shown in
FIG. 3A, however, including also an additional link memory unit 50, for
the purpose of fully and uniquely representing I/O code-word inputs of any
length. Links L1 and L2 are mandatory links used for any input, L2 being
the main link as in all above described embodiments. In FIG. 5A is shown
the same set of eleven previously stored EXs as in FIG. 3A and the current
EX input is 'et cetera'. In FIG. 5B the EX 'et cetera' has been added and
in FIG. 5C previous and current input conditions are as in FIG. 5A.
An output in form of an EX match signal and/or an EXS value is generated
if an overall link value matching condition has been produced. The EXS
value is derived from the relative address of the main link storage
location where the current main link value is found. In analogy with the
embodiments of FIG. 1E & 1F, pointer address values may be output from a
second column holding such values within each main link table T2m.
In FIG. 5A code conversion means 46 append at the end of string 32 a
terminator in form of a control code-word with an alphet value 1™.
The addition of 'et cetera' in FIG. 5B serves the purpose to explain how
proper sorting order is achieved also when additional links are employed.
To represent compounded EXs and EXs with capitals a two alphet sequence is
used as a terminator, starting with the system control code-word 0™.
The purpose of FIG. 5C is to demonstrate the advantages of incorporating
bit map memory means at the front end of the additional link memory unit 50.
The additional link memory unit 50 shown in FIG. 5A & 5B comprises in
itself a series of small EX systems, here employed as sub-systems. The
first such sub-system in the series uses as input the EXS code-word
concatenated with the third link alphet, transformed into two digital
words. The second one of those two digital words comprises the last three
bits from the EXS code-word combined with the third link alphet. The first
digital word is added to a base address B3o and the resulting address is
used to locate and read out a pair of boundary addresses from a boundary
address table having its first row at address B3o. This first sub-system
produces as an output a sequence-number T3S. The next sub-system in the
series uses the T3S code-word concatenated with the next link alphet as an
input which is transformed into two new digital words. The functioning is
iterated until there are no more link alphits left.
Unless a match has been found, an overall link value matching condition
is looked for by test of alternative overall additional link values,
should previously stored link values have generated such alternatives.
Such alternatives may or may not relate individually to duplicated main
link storage locations holding identical current main link values. The
case of such duplication is shown in FIG. 5B, where storage of the
additional EX 'et cetera' has caused a L2™ value to be stored within the
main link memory unit, a value identical with the L2™ value representing
the EX 'et cetera'. By comparing the contents of string 32 in FIG. 5A and
FIG. 5B it can be seen that the open compound 'et cetera' is stored
exactly as the corresponding solid compound 'etcetera', except for the terminators that are designed to adjust the least significant part of the overall link value to facilitate a proper sorting order. In case one or more capital letters are part of an EK, each capital positioning alternative is represented by means of a specially selected L8 value. As shown in FIG. 5B the overall link value matching condition has been produced already upon examining the first one of the two boxes holding the value $E9^x$, i.e. number 4 in the sequential search of $L2_{mn}$ values. $EKS=9$ is derived from the relative address of the main link storage location related to this first matching alternative, i.e. the $L2_{mn}$ location.

In FIG. 5C the bit map memory means 52 use as an input the EKS output produced from the main link memory unit 14. Hence, $S1= EKS$. The functioning of the map is equivalent to what has been explained with reference to FIG. 3C. However, in FIG. 5C is shown just one large map accommodating $WWW^x$ bit positions, i.e. $WWW^x$ EKS values. The map output $M1$ is an ordinal within a set comprising all EKs that require more than 4 alphits. Only two of the eleven code-words that are presumed to have been previously stored have caused a bit mark within the map as the others require 4 or less alphits to be fully identified. In a search operation the bit map is always tested for a bit mark condition. In case the relevant bit position is not marked, this is an indication that no tail to the four first alphits has been stored – no time is wasted consulting an additional link memory unit like the one discussed with reference to FIG. 5A & 5B.

As will be shown below with reference to other figures, the additional link memory unit may take effect at a later point in the I/O code-word sequence.

FIG. 6A through 6E relate to embodiments including intermediate links. Code conversion to alphit code-words is employed. FIG. 6A illustrates a basic concept applied in all the embodiments shown in FIG. 6B through 6E.

Following conventions are used for designating values, addresses, tables et cetera:

- $p$ is an ordinal index $(0, 1, 2, \ldots)$ designating all previously stored boundary address value pairs, B2 & B4, in ascending address value order. Each B2 and B4 value represents the address of the first row of a table, address $B2_o$ pointing at table $T2_o$ located within the intermediate link memory unit and address $B4_{000}$ pointing at table $T4_{000}$ within the main link memory unit.

- $q$ is an ordinal index $(0, 1, 2, \ldots)$ for each particular value of $p$, designating previously stored $L2_m$ values in ascending value order as $L2_{mn}$ values.
r is an ordinal index \((0,1,2,...)\) for each particular value of pq, designating previously stored \(L_{3pq}\) values in ascending value order as \(L_{3pq}\) values.

s is an ordinal index \((0,1,2,...)\) for each particular value of pqr, designating previously stored \(L_{4pqr}\) values in ascending value order as \(L_{4pqr}\) values.

Previously stored \(L_4\) values are held in one functionally contiguous string within the main link memory unit 14, each such designation carrying a pqr index when shown in a figure. In FIG. 6A values \(L_{4pqr}\) through \(L_{4pqr}\) are stored in value ascending order with ascending address values. As this convention is being followed for all pqr groups the series of composite pqr numbers are forming a number series in ascending pqr value order with ascending address values. The embodiment of FIG. 6A include input means 10, having eight current alpht values \(A_0-A_7\) within string 32 and four current link values \(L_1-L_4\) within string 34. A first link memory unit 12 provides current boundary address values \(B_{2n}\) and \(B_{4n,0n}\), related to the current first link value \(L_1\). A main link memory unit 14 holds in store previously stored main link values \(L_{4pqr}\). The intermediate link memory unit 58 has a plurality of storage locations functionally arranged in two column relation tables, 60 & 62. These tables hold value pairs, each comprising a link value and a related address component value. The value pairs are accommodated in two sets of tables, one set of \(T_{2n}\) tables 60 and one set of \(T_{3n}\) tables 62, each table holding one or several value pairs in separate rows thereof. A current second link boundary address value \(B_{2n}\) is used to locate a current second link table and the link values \(L_{2pq}\) held within such current second link table are read out for comparison with the current second link value \(L_2\). An \(L_2\) value matching condition is produced if a previously stored identical second link value \(L_{2pq}\) is found and an address value is produced, pointing at a current third link table containing all third link values \(L_{3pqr}\) having previously been put in store as related to a previously stored second link value \(L_2\). The address value is produced using related address component values \(C_{2n}\) read out from the second link table 60 for determining an off-set value relative the location of the second link table 60. The link values \(L_{3pqr}\) held within such current third link table are read out for comparison with the current third link value \(L_3\) and an \(L_3\) value matching condition is produced if a previously stored identical third link value \(L_{3pqr}\) is found. A current main link table 64 can now be located, i.e. a table that holds all main link values \(L_{4pqr}\) having previously been put in store as related to
the previously stored third link value L₃ₘᵢₘᵢₜ. Current boundary address values B₄ₘᵢₜ and B₄ₘᵢᵣ are produced by adding to the current boundary address B₄ₘᵢ₀ off-set values, each such off-set value being determined from one or more address component values C₃ₘᵢᵣ, read out from the current third link table. The link values L₄ₘᵢᵣ that are held within such current fourth link table, are read out for comparison with the current fourth link value L₄ and an L₄ value matching condition is produced if a previously stored identical main link value L₄ₘᵢᵣ is found. Table 64 is searched sequentially according to the alphabetic numbering shown in FIG. 6A. At number 1ₘₘ the match is obtained and the related pointer address P₃ₘᵢᵣ is output.

FIG. 6B & 6C illustrate two arrangements using different methods for holding in store previously stored address component values within the intermediate link unit 58. Hence different procedures are followed for determining the off-set values. The I/O code-word input represents a sequence formed by the eight most frequently occurring letter symbols of the English alphabet, 'etaonrh'. It is presumed that previously stored link values and address component values represent all possible eight symbol combinations of these eight specific letter symbols, corresponding to 16,777,216 different EKs. The main link memory unit 14 comprises a string of almost 17 million byte storage locations holding a corresponding number of L₄ₘᵢᵣ values, along with a string holding related pointer address values. To conveniently cover the wide address range of these strings addressing by means of an off-set from the current boundary address B₄ₘᵢ₀ is employed as described above with reference to FIG. 6A. Both arrangements produce, using different procedures, identical off-set address values relative the B₄ₘᵢ₀ address value.

FIG. 6B shows how a match between L₂ and L₂ₘᵢᵣ is obtained after a binary search within table 60, comprising the four steps numbered 0, 1, 2 & 3. From the second column are read out all address component values held within rows 0 through 9 and these C₃ₘᵢᵣ values are added, as the integral sign is meant to illustrate. The sum, used as a relative address value, points at table T₃ₘᵢᵣ wherein a value L₃ₘᵢᵣ, matching the current value L₃, is found after four step binary search. The B₄ₘᵢᵣ value, equivalent to the B₄ₘᵢ₀ value, can now be produced by a summation of C₃ₘᵢᵣ values from C₃ₘᵢ₀ through C₃ₘᵢ₀₀ and including B₄ₘᵢ₀ in the summation. B₄ₘᵢ₀₀ is produced by excluding C₃ₘᵢ₀₀ from the summation. Boundary address value B₂ₘᵢᵣ was used to determine where to start the first binary search and where to start the summation of C₂ₘᵢᵣ values. The value B₃ₘᵢᵣ is used only in case table T₃ₘᵢᵣ is involved, determining the end of the T₃ₘᵣ searching range. As shown in
FIG. 6B as well as in FIG. 6C C2_0 and C2_1 values are used to determine the ends of respective current searching ranges.

FIG. 6C shows 16-bit wide second columns within tables 60 & 62, introduced in order to avoid the two summations described above with reference to FIG. 6B. The location of table T3_0 is thereby found much faster by direct use of values C2_0 and C2_1 as pointers relative table T2_0. Further, values C3_0 and C3_1 are used directly as offset values to be added to value B4_0000, whereby the output pointer P_business is found with minimal delay. As can be seen from FIG. 6B as well as from FIG. 6C, the binary search technique is used also within table 64 in order to avoid unnecessary delays.

FIG. 6D shows bit map memory means used as a sub-system comprising part of code conversion means. The bit map sub-system facilitates elimination of all redundant boundary address storage locations within the first link memory unit 12. The set of L1_0 values that are represented within memory unit 12 is compressed into a small set of ordinal code-words. The highest possible L1_0 value for the embodiments shown in FIG. 6A through 6D will thereby be reduced from 65535 to a substantially lower value, depending on the characteristics of the EX set having been previously stored. With the assumptions made above with reference to FIG. 6B & 6C, the highest L1_0 value is reduced to 1023, whereby first link storage requirements are reduced by a factor 64. The bit map memory would require a 256 byte T1 table, a 64 byte TM1 table, a 512 byte T2 table, a 128*2 byte TM2 table and the small ordinary TM table 54.

In FIG. 6D it is presumed that the same eleven EXs as shown in FIG. 3A have been previously stored, one of them, i.e. 'et cetera', being used for current input. For this case the bit map memory requirements are the same except for table T2 and TM2, now reduced to 24 and 6 bytes respectively. The detailed functioning of the bit map memory system has been illustrated and described in relation to FIG. 3C. The value of the ordinal code-word referred to above, being assigned as the current link value L1, is equivalent to the EX output of the bit map sub-system. Hence L1 is produced as a count of the number of marked bits within table T2 at bit addresses below the relative bit address 05E0, shown in FIG. 6D to represent the current input. As can be concluded from FIG. 6D

L1=M2_0+M2_1+M0_0=7. Without addition of a bit map sub-system L1 would equate (9*32*32+24*32+7)*2=19982.

FIG. 6E shows an embodiment coping with input sequences generating fourteen alphit values. A possible additional link unit is not shown in FIG. 6E because of space limitations. Code conversion is made in two steps
like in FIG. 6D, in FIG. 6E illustrated as being performed within first
input means 66 and second input means 68, respectively. Within second
input means 68 two EX systems in parallel are employed as sub-systems for
producing ordinal code-word values, i.e. EXS values, which, when linked
together within string 32, uniquely represent the current EX input.

FIG. 7A - 7B relate to EX system embodiments with input means employing
code conversion to 4-bit nibble code-words and variable length code-words,
respectively. The two embodiments both have code conversion means
facilitating compact representation of EX prefix attributes as well as EX
suffix attributes.

FIG. 7A illustrates how an input code-word sequence representing the EX
'etching' is converted to a nibble sequence 3D1569F1x, all in accordance
with the code schema shown in FIG. 2B. The code conversion control program
also executes a search for a compact code-word representation of EX prefix
attributes within look-up table 70 and of EX suffix attributes within
look-up table 72. The algorithm is shown in the lower left corner of the
figure. The first four nibble values 3D15x are fed as a search argument to
the prefix table and a value E is returned indicating the number of
nibbles that have been allocated a compact nibble value Na. The E=0
response means no prefix found and Na is returned with the value 0. A
value U=4 is returned from the suffix table along with Na=1, indicating
that the value Na=1 shall substitute the last four nibbles of the search
argument, i.e. the whole of 69F1x, which represents the suffix 'ing'. The
four nibble values 3D15x are allocated to Na through N7 and remaining
nibble pairs not required to represent the EX input sequence are allocated
a system control code-word value FDx, representing a nullity. Any singular
unused nibble would be located in position 7 and given the value Fx, also
representing a nullity when placed in the last position prior to the
nibbles used to represent prefixes and suffixes. The FDx value is also
used as an additional link terminator.

FIG. 7B illustrates how an input code-word sequence representing the EX
'etcetera' is converted to a variable length code-word representation in
accordance with the code schema shown in FIG. 2C. No prefix or suffix is
found, nevertheless the eight symbol EX can easily be represented by the
first three link values. The last two bit positions prior to the nibble
positions used to represent prefixes and suffixes are filled up with 11x,
any sequence of binary ones representing a nullity in this position.

With reference to FIG. 8A, FIG. 7A, FIG. 2C and FIG. 1C the functioning
of a preferred embodiment of a file handling system for storing and
transferring compressed text files will be explained.
FIG. 8A relates to a text file handling system embodiment employing variable length code-words. In the following such file handling system will be referred to as an FH system for short. FIG. 8A shows the schema used for encoding and decoding textual words, i.e. ordinary graphic words as they appear within a running text. The text is input to the FH system in form of long sequences of standard I/O code-words, e.g. octet I/O code-words according to EASCII standard.

Each sub-sequence inbetween space separators, i.e. a textual word, is identified by the FH system as an entity, in the following referred to as a file segment. Capitalization of letters and also special signs at the beginning and end of a textual word, e.g. quotation marks, punctuation marks and parenthesis, are taken care of separately using techniques falling outside the scope of the embodiment being described.

For the particular type of text files to be handled by the FH system, e.g. ordinary english texts for general use, the expected frequencies of occurrence have been determined for all frequent segments. The 80 most common segments have been allocated 8-bit RSS code-words $10^8-5F^8$ as is shown in FIG. 8A. Also shown are 1280 12-bit RSS code-words for allocation to less frequently occurring frequent segments and 20480 16-bit RSS code-words for allocation to least frequently occurring frequent segments.

Each of the three RSS code-word groups is represented within the embodiment by an EX system embodiment according to FIG. 7A, in the following referred to as segment reference means. Each such segment reference means provides an EXS code-word, in the following named RSS code-word, in response to any valid segment input, i.e. an input equivalent to a previously stored EX. The FH system inputs each file segment in a prescribed order to the three segment reference means until a valid RSS code-word has been produced. In case none of the three segment reference means provides such a response, the FH system selects the 4-bit RSS code-word $0^8$, which means that a particular number of succeeding code-words shall be encoded and decoded according to the code schema shown in FIG. 2C, i.e. a variable length code capable of encoding and decoding any unfrequent file segment on a symbol by symbol basis. As is indicated in FIG. 8A, immediately following the RSS code-word $0^8$ is one more 4-bit code-word determining the number of code-words to follow, representing individual unfrequent segment symbols.

All FH code-words have the prefix-free property and may be concatenated in strings of any length when held in store or being transmitted. Single spaces between segments does not require any special representation.

Decoding is performed using the embodiment according to FIG. 7A, however, with the control program set to operate in a reversed mode, basically as has been described with reference to FIG. 1C.
B. CLAIMS

1. A system for storing a set of entity keys and for providing a response to a subsequent input of an individual entity key code representation, comprising

input means for transforming part of a current entity key I/O code-word input into two or more linked digital words, which thereby each attains a current link value and jointly a current overall link value uniquely representing such part of such I/O code-word input,

memory means for holding in store link values and address component values related to previously stored entity keys, and

control means connected to said input means and to said memory means for determining a current address range for each link succeeding the first one, using at least one current link value to find previously stored address component values to determine the boundaries of such current range, for comparing the current value of each link succeeding the first one with at least one link value previously stored within such current address range, and for producing (i) an overall link value matching condition if, for each link succeeding the first one, the current link value matches a previously stored link value or (ii) an overall link value non-matching condition if the current overall link value is found to fall inbetween overall link values related to previously stored entity keys.

2. A system as defined in claim 1, having storage locations within said memory means provided by read/write type memory devices, thereby facilitating storage of previously not stored entity key I/O code-word inputs, wherein said control means include

means for presenting a first link reject signal, should the current first link value not generate a succeeding link address range,

means for presenting a succeeding link reject signal, should a current succeeding link value not match any previously stored link value within the current address range,

means for executing write instructions in response to a first link reject signal, storing within said memory means, at locations related to the current first link value, address component values essential for determining new address ranges, at least for the next link, inserting within such new address range, for each one of all succeeding links, the current succeeding link value and, except for the last link, storing
address component values related to each inserted link value, essential for determining new address ranges, at least for the next link.

means for executing write instructions in response to a succeeding link reject signal, inserting within said memory means, at a location within such current address range where a current link value turned out not to match any previously stored link value, such non-matching current link value, also inserting within a new address range, for each one of any remaining succeeding links, the current succeeding link value and, except for the last link, storing address component values related to each inserted link value, essential for determining new address ranges, at least for the next one of any remaining succeeding links, and means for executing move instructions prior to the insertion of current link values, thereby providing accommodation for such current link values as well as for any related address component values and for adjusting previously stored address component values to conform to any address changes resulting from the execution of such move instructions.

3. A system as defined in claim 1, wherein said control means include means for outputting, if the overall link value matching condition has been produced, an address component value previously put in store within said memory means as uniquely related to such overall link value.

4. A system as defined in claim 1, wherein said control means include means for outputting, if an overall link value non-matching condition has been produced, an address component value previously put in store as related to an interval including the current overall link value, inbetween adjacent overall link values representing previously stored entity keys.

5. A system as defined in claim 1, wherein said input means include code conversion means for substituting one or more compact code-words for the entity key I/O code-word input, each compact code-word being drawn from a compact code, such compact code not necessarily being the same for each individual position within a succession of compact code-words, for temporarily allocating the overall link value of the compact code-word representation to functionally successive memory cells, and for assigning the contents of such successive memory cells to the linked digital words, not excluding, prior to such assigning, a substitution of one or more ordinal code-words for the contents of one or more sections of such functionally successive memory cells.
6. A system as defined in claim 1, wherein said input means include code conversion means for producing one ordinal code-word to replace and represent at least one section of a string representation of part of the entity key I/O code-word input, for iteratively producing at least one further ordinal code-word to replace and represent a concatenated digital word formed by linking, in any prescribed order, the ordinal code-word just having been produced with at least one more section of such string, for temporarily allocating the overall value of a last concatenated digital word to functionally successive memory cells, such last digital word being formed by linking, in any prescribed order, the ordinal code-word just having been produced with any remaining sections of the string, and for assigning the contents of such successive memory cells to at least one of the linked digital words.

7. A system as defined in claim 1, wherein mandatory links, i.e. a fixed minimum number of links, are used for any input, the last one of which is a main link, wherein at least one additional link is required to fully and uniquely represent such I/O code-word inputs that generate more digits than can be accommodated within the mandatory links, and wherein said input means include means for inserting, within at least one of the mandatory links, specific digits representing a nullity as far as the entity key I/O coding is concerned, should the entity key I/O code-word input not generate sufficient number of digits for the mandatory links to attain defined link values, and wherein said control means include means for outputting, if the overall link value matching condition has been produced, an entity key reference number derived from the relative address of the main link storage location where the current main link value was found.

8. A system as defined in claim 1, wherein mandatory links, i.e. a fixed minimum number of links, are used for any input, the last one of which is a main link, wherein at least one additional link is required to fully and uniquely represent such I/O code-word inputs that generate more digits than can be accommodated within the mandatory links, and wherein said memory means include bit map memory means for holding in store a bit map representation of the set of main link reference numbers that relate to previously stored I/O code-word inputs requiring such additional links.
9. A system as defined in claim 7, wherein said memory means include means for producing the overall link value matching condition by examining alternative overall additional link values, should previously stored link values have generated such alternatives in order to uniquely represent different inputs generating identical overall mandatory link values, and wherein said control means include means for outputting, if the overall link value matching condition has been produced upon such examining, a two part entity key reference number, a main part derived from the relative address of the storage location where the current main link value was found and an additional part uniquely identifying the matching overall additional link value.

10. A system as defined in claim 7, wherein said memory means include means for producing the overall link value matching condition by examining alternative overall additional link values, should previously stored link values have generated such alternatives individually related to duplicated main link storage locations holding identical current main link values, and wherein said control means include means for outputting, if the overall link value matching condition has been produced upon such examining, an entity key sequence number derived from the relative address of the particular main link storage location being related to the matching alternative.

11. A system as defined in claim 10, having means for generating an entity key I/O code-word output in response to an input of a code representation of a current entity key sequence number, wherein said control means include means for determining the current absolute address of a particular main link storage location, the current relative address of which equals the current entity key sequence number, and for reading from such particular storage location a current main link value, means for searching said memory means for a current main link address range that is encompassing the current absolute address, and for producing current preceding link values as related to the current main link address range, as well as any current additional link values as related to the current main link absolute address, and output means for retransforming the overall link value of the complete series of current link values to a corresponding entity key I/O code-word output.
12. A system as defined in claim 1, transforming an entity key I/O code-word input into at least three digital words, wherein said memory means include
a first link memory unit for providing current boundary addresses
related to current first link values,
a main link memory unit for holding in store previously stored main link values,
the intermediate link memory unit, having a plurality of storage locations functionally arranged in two column relation tables for holding value pairs, each such pair comprising a link value and a related address component value, for accommodating the value pairs in one or several sets of tables, one set for each intermediate link, each table holding one or several value pairs in separate rows thereof, and for using at least one current second link boundary address to locate a current second link table,
and wherein said control means include
means for comparing the link values held within such current second link table with the current second link value, for producing a second link value matching condition if a previously stored identical second link value is found, and for computing an address value pointing at a current third link table containing all third link values having previously been put in store as related to a previously stored second link value, the address value being produced using related address component values extracted from the current second link table for computing an off-set value relative the location of the current second link table or relative a current third link boundary address being obtained from said first link memory unit, for comparing the link values held within such current third link table with the current third link value, for producing a third link value matching condition if a previously stored identical third link value is found, and, provided the link in question is not the main link, an address value pointing at a current fourth link table containing all fourth link values having previously been put in store as related to a previously stored third link value, the address value being produced using related address component values extracted from the current third link table, not excluding the alternative of extracting address component values from preceding third link tables within a sub-set of third link tables related to the current second link table, for computing an off-set value relative the location of the current third link table, relative the location of the current second link table or relative a current fourth link boundary address being obtained from said first link memory unit, and
for repeating such functions in an analogous manner link by link until the
main link is supposedly being reached.

13. A system as defined in claim 5, using compact code-words to
represent lowercase letters, preferably having been allocated ascending
values in progressing alphabetic order, wherein said code conversion
means include

means for including at the end of the succession of compact code-words
a control function code-word to make such succession represent also
capital letters, such control function code-word comprising at least one
bit and having one value, preferably the highest, when governing an
all-lowercase letter sequence and various other values, preferably lower,
for alternative capital letter constellations, in particular an extreme
value to make the sequence represent an all-capital entity key, whereby
such entity key will be positioned correctly within a sequence of entity
keys being sorted in overall link value order, and/or

means for including at the end of the succession of compact code-words a
control function code-word to make such succession represent also an open
or hyphenated compound, such control function code-word having one value,
preferably the highest, when governing a solid compound and other values
when governing an exact space or hyphen position, in particular an extreme
value to make the sequence represent an open compound with a single
letter as the first element, such as 'a priori', and a value next to
such an extreme value to make the same succession of compact code-words
represent the corresponding hyphenated compound, i.e. 'a-priori', or

means for including at the end of the succession of compact code-words
a control function code-word to make such succession represent also
capitalized compounds, still preserving a desirable sorting order by
utilizing a wide spectrum of control function code-word values, using an
extreme value to govern an all-capital compound open after the first
letter, such as 'A PRIORI'.

14. A system as defined in claim 5, wherein said code conversion means
include

means for generating a compact code representation, whereby compact-
code-words representing frequently occurring entity key I/O code-words,
pairs of entity key I/O code-words or larger groups of entity key I/O
code-words are made up of fewer bits and in case of less-frequent
occurrences of larger number of bits.
15. A system as defined in claim 5, wherein said code conversion means include
means for holding in store a compact code-word representation of at least one entity key attribute set, each such set representing a unique selection of entity key attributes, and
means for comparing a succession of compact code-words with the compact code-word representation of at least one entity key attribute set and for substituting one specific compact code-word for the specific part of such succession of compact code-words that is equalling the compact code-word representation of a specific entity key attribute, such specific compact code-word being drawn from a compact code being unique for the individual code-word position within the refreshed succession of compact code-words or being unique as governed by a code shift control code-word.

16. A system for storing a set of entity keys and for providing a response to a subsequent input of an individual entity key code representation, comprising
input means for transforming part of a current entity key I/O code-word input into one digital word comprising at least one section, whereby each such section attains a current section value and when linked together a current overall section value uniquely representing such part of such I/O code-word input,
bit map memory means for holding in store bit representations of section values related to previously stored entity keys, and
control means connected to said input means and to said memory means for finding a current bit location using a current first section value as a relative address value, for reading out the bit value of such location to test if a previously stored entity key has produced a bit mark or not, for producing, upon a bit mark match, a section ordinal code-word from a count of the number of marked bit locations within a bit location address range below or above the address of the current bit location, for outputting an input response if no current section value remains unused, else producing a further relative address value by concatenating, in any prescribed order, the current contents of at least one remaining section with the preceding section ordinal code-word just having been produced,
for finding, within an individually section related part of the bit map, a further current bit location using such further relative address value and for repeating iteratively the functioning, section by section, as long as any current section value remains unused and the bit mark match is produced.
17. An arrangement as defined in claim 16, wherein said control means include means for producing marked bit counts by reading out overall binary word values from consecutive bit map sub-areas, each comprising a succession of bit locations, and for using such overall values as entries into a look-up table providing sub-area counts as outputs.

18. A system as defined in claim 16, wherein said memory means include means for holding in store, at least within one section related part of the bit map, a set of marked bit counts, each member of such set representing a count taken within a limited address range, for providing, related to a bit location close to the current bit location, an aggregate count value produced from at least one such limited address range count, and for producing the current section ordinal code-word from such aggregate count value, adding or subtracting a marked bit count produced for the interval between the close and the current bit location.

19. A file handling system for storing and transferring compressed files, each file containing at least one segment, any such segment being either a frequent segment or an unfrequent segment, each segment being input to and output from the system encoded into I/O code-words drawn from a distinct I/O code, comprising segment reference means for holding at least one reference segment data set, each such reference segment data set comprising a system representation of a special selection of frequent segments, and for allocating a distinct RSS code-word to each reference segment, such distinct RSS code-word being retrievable for any frequent segment, using the frequent segment as a key, and conversely, any frequent segment being retrievable using a distinct RSS code-word as a key, coding means for substituting one distinct RSS code-word for each distinct sequence of I/O code-words that is representing a distinct frequent segment, each such distinct RSS code-word being retrieved from a specific reference segment data set and applied inside the file handling system as a file code-word, and for substituting a distinct sequence of file code-words for each sequence of I/O code-words that is representing a distinct unfrequent segment, such file code words being drawn from a distinct variable length type of code using fewer digits to represent frequently occurring I/O code words, pairs of I/O code-words or larger groups of I/O code-words and a larger number of digits in cases of less frequent occurrences, and
decoding means for outputting an I/O code-word representation of a file being retrieved from the system storage or being transferred from another system location, thereby substituting I/O code-words for the file code-words.

20. A file handling system as defined in claim 19, wherein said coding means include
means for employing variable length RSS code-words to represent frequent segments, using fewer digits to represent more frequently occurring frequent segments and a larger number of digits in case of a less frequently occurring frequent segments.

21. A method for storing a set of entity keys and for providing a response to a subsequent input of an individual entity key code representation, including the steps of
transforming part of a current entity key I/O code-word input into at least two linked digital words, which thereby each attains a current link value and jointly a current overall link value uniquely representing such part of such I/O code-word input,
determining a current address range for each link succeeding the first one, using at least one current link value to find previously stored address component values to determine the boundaries of such current range,
comparing the current value of each link succeeding the first one with at least one link value previously stored within such current address range,
and producing (i) an overall link value matching condition, if for each link succeeding the first one, the current link value matches a previously stored link value or (ii) an overall link value non-matching condition if the current overall link value is found to fall inbetween overall link values related to previously stored entity keys.
AMENDED CLAIMS

[received by the International Bureau on 21 January 1988 (21.01.88)
original claim 4 amended; claims 1-3, 5 and 6 unchanged; claim 7 replaced by amended
claims 7 and 8; claims 8-10 replaced by amended claims 10, 13 and 14; claims 11-21
amended and renumbered as claims 15-25; new claims 9, 11 and 12 added (10 pages)]

1. A system for storing a set of entity keys and for providing a
response to a subsequent input of an individual entity key code
representation, comprising
input means for transforming part of a current entity key I/O code-word
input into two or more linked digital words, which thereby each attains a
current link value and jointly a current overall link value uniquely
representing such part of such I/O code-word input,
memory means for holding in store link values and address component
values related to previously stored entity keys, and
control means connected to said input means and to said memory means
for determining a current address range for each link succeeding the
first one, using at least one current link value to find previously
stored address component values to determine the boundaries of such
current range, for comparing the current value of each link succeeding
the first one with at least one link value previously stored within such
current address range, and for producing (i) an overall link value
matching condition if, for each link succeeding the first one, the
current link value matches a previously stored link value or (ii) an
overall link value non-matching condition if the current overall link
value is found to fail between overall link values related to
previously stored entity keys.

2. A system as defined in claim 1, having storage locations within said
memory means provided by read/write type memory devices, thereby
facilitating storage of previously not stored entity key I/O code-word
inputs, wherein said control means include
means for presenting a first link reject signal, should the current
first link value not generate a succeeding link address range,
means for presenting a succeeding link reject signal, should a current
succeeding link value not match any previously stored link value within
the current address range,
means for executing write instructions in response to a first link
reject signal, storing within said memory means, at locations related to
the current first link value, address component values essential for
determining new address ranges, at least for the next link, inserting
within such new address range, for each one of all succeeding links, the
current succeeding link value and, except for the last link, storing
address component values related to each inserted link value, essential for determining new address ranges, at least for the next link,
means for executing write instructions in response to a succeeding link reject signal, inserting within said memory means, at a location within such current address range where a current link value turned out not to match any previously stored link value, such non-matching current link value, also inserting within a new address range, for each one of any remaining succeeding links, the current succeeding link value and, except for the last link, storing address component values related to each inserted link value, essential for determining new address ranges, at least for the next one of any remaining succeeding links, and means for executing move instructions prior to the insertion of current link values, thereby providing accommodation for such current link values as well as for any related address component values and for adjusting previously stored address component values to conform to any address changes resulting from the execution of such move instructions.

3. A system as defined in claim 1, wherein said control means include means for outputting, if the overall link value matching condition has been produced, an address component value previously put in store within said memory means as uniquely related to such overall link value.

4. A system as defined in claim 1, wherein said control means include means for outputting, if an overall link value non-matching condition has been produced, at least one address component value previously put in store as related to an interval that (i) includes the current overall link value and (ii) is defined by a pair of adjacent overall link values representing two previously stored entity keys.

5. A system as defined in claim 1, wherein said input means include code conversion means for substituting one or more compact code-words for the entity key I/O code-word input, each compact code-word being drawn from a compact code, such compact code not necessarily being the same for each individual position within a succession of compact code-words, for temporarily allocating the overall link value of the compact code-word representation to functionally successive memory cells, and for assigning the contents of such successive memory cells to the linked digital words, not excluding, prior to such assigning, a substitution of one or more ordinal code-words for the contents of one or more sections of such functionally successive memory cells.
6. A system as defined in claim 1, wherein said input means include code conversion means for producing one ordinal code-word to replace and represent at least one section of a string representation of part of the entity key I/O code-word input, for iteratively producing at least one further ordinal code-word to replace and represent a concatenated digital word formed by linking, in any prescribed order, the ordinal code-word just having been produced with at least one more section of such string, for temporarily allocating the overall value of a last concatenated digital word to functionally successive memory cells, such last digital word being formed by linking, in any prescribed order, the ordinal code-word just having been produced with any remaining sections of the string, and for assigning the contents of such successive memory cells to at least one of the linked digital words.

7. A system as defined in claim 1, wherein at least two dedicated links are employed at any said entity key I/O code-word input, wherein the last one of such dedicated links is a main link, and wherein said control means include means for providing as an output a main link sequence-number within a set comprising all relative addresses of previously stored main link values, such sequence-number being equivalent to the ordinal that represents the particular relative address where a previously stored main link value, matching the current main link value, is found.

8. A system as defined in claim 7, and wherein the input means include means for inserting, within at least one of said dedicated links, specific digits representing a nullity as far as the entity key I/O coding is concerned, should the entity key I/O code-word input not generate sufficient number of digits for such links to attain defined link values.

9. A system as defined in claim 7, wherein a series of additional link sub-systems are employed, each such sub-system functioning as an entity key system, wherein the input means of a first such sub-system include code conversion means for producing internal sub-system link values using as input (i) the main-link sequence-number output and (ii) the first additional link value, wherein the input means of any succeeding sub-system include code conversion means for producing internal sub-system link values
using as input (i) the preceding sub-system sequence-number output and
(ii) the next additional link value, and
wherein the control means include
sub-system output means for providing said sub-system sequence-number
output, and
system output means for providing (i) an overall link value matching
condition if a previously stored I/O code-word input has caused internal
sub-system link values to be stored, identical with those generated by
the current I/O code-word input, or (ii) an overall link value
non-matching condition.

10. A system as defined in claim 7, wherein at least one additional
link is employed to facilitate an unambiguous representation of all
members of a set of entity keys, and wherein the memory means include
bit map memory means for holding in store a bit mark representation of
a sub-set of main link sequence-numbers, each member of such sub-set
representing a previously stored entity key I/O code-word input that has
generated at least one additional link value.

11. A system as defined in claim 10, wherein the control means include
bit map control means connected to said bit map memory means for
finding the specific bit location that represents the current main link
sequence-number, for reading the bit value of such location to test if a
previously stored entity key I/O code-word input has generated a bit mark
at said specific bit location, for producing, upon a bit mark match, an
ordinal code-word from a count of marked bit locations within a bit
location address range below or above the address of the current bit
location, and for providing such ordinal code-word, i.e. a member of a
zero redundancy sub-set of main link sequence-numbers, as an output to a
succeeding additional link sub-system.

12. A system as defined in claim 7, wherein memory means include
an additional link main memory unit for storing a first additional link
value, and
wherein code conversion means include
means for providing a sequence-number code-word representing any
additional link value sequence, and for substituting such code-word for
said first additional link value, facilitating storage within said
additional link main memory unit of such sequences of any length.
13. A system as defined in claim 7, wherein said memory means include means for producing the overall link value matching condition by examining alternative overall additional link values, should previously stored link values have generated such alternatives in order to uniquely represent different inputs generating identical overall dedicated link values.

and wherein said control means include means for outputting, if the overall link value matching condition has been produced upon such examining, a two part entity key reference number, a main part derived from the relative address of the storage location where the current main link value was found and an additional part uniquely identifying the matching overall additional link value.

14. A system as defined in claim 7, wherein said memory means include means for producing the overall link value matching condition by examining alternative overall additional link values, should previously stored link values have generated such alternatives individually related to duplicated main link storage locations holding identical main link values.

and wherein said control means include means for outputting, if the overall link value matching condition has been produced upon such examining, an entity key sequence-number derived from the relative address of the particular main link storage location being related to the matching alternative.

15. A system as defined in claim 10, having means for generating an entity key I/O code-word output in response to an input of a code representation of a current entity key sequence number, wherein said control means include means for determining the current absolute address of a particular main link storage location, the current relative address of which equals the current entity key sequence-number, and for reading from such particular storage location a current main link value.

means for searching said memory means for a current main link address range that is encompassing the current absolute address, and for producing current preceding link values as related to the current main link address range, as well as any current additional link values as related to the current main link absolute address, and output means for retransforming the overall link value of the complete series of current link values to a corresponding entity key I/O code-word output.
16. A system as defined in claim 1, transforming an entity key I/O
code-word input into at least three digital words, wherein said memory
means include
a first link memory unit for providing current boundary addresses
related to current first link values,
a main link memory unit for holding in store previously stored main
link values,
an intermediate link memory unit, having a plurality of storage
locations functionally arranged in two column relation tables for holding
value pairs, each such pair comprising a link value and a related address
component value, for accommodating the value pairs in one or several sets
of tables, one set for each intermediate link, each table holding one or
several value pairs in separate rows thereof, and for using at least one
current second link boundary address to locate a current second link table,
and wherein said control means include
means for comparing the link values held within such current second link
table with the current second link value, for producing a second link
value matching condition if a previously stored identical second link
value is found, for computing an address value pointing at a current
third link table containing all third link values having previously been
put in store as related to a previously stored second link value, the
address value being produced using related address component values
extracted from the current second link table for computing an off-set
value relative the location of the current second link table or relative a
current third link boundary address being obtained from said first link
memory unit, for comparing the link values held within such current third
link table with the current third link value, for producing a third link
value matching condition if a previously stored identical third link value
is found, and, provided the link in question is not the main link, an
address value pointing at a current fourth link table containing all
fourth link values having previously been put in store as related to a
previously stored third link value, the address value being produced using
related address component values extracted from the current third link
table, not excluding the alternative of extracting address component
values from preceding third link tables within a sub-set of third link
tables related to the current second link table, for computing an off-set
value relative the location of the current third link table, relative the
location of the current second link table or relative a current fourth
link boundary address being obtained from said first link memory unit, and
for repeating such functions in an analogous manner link by link until the
main link is supposedly being reached.

17. A system as defined in claim 5, using compact code-words to
represent lower-case letters, preferably having been allocated ascending
values in progressing alphabetic order, wherein said code conversion
means include

means for including at the end of the succession of compact code-words
a control function code-word to make such succession represent also
capital letters, such control function code-word comprising at least one
bit and having one value, preferably the highest, when governing an
all-lower-case letter sequence and various other values, preferably
lower, for alternative capital letter constellations, in particular an
extreme value to make the sequence represent an all-capital entity key,
whereby such entity key will be positioned correctly within a sequence of
entity keys being sorted in overall link value order, and/or

means for including at the end of the succession of compact code-words a
control function code-word to make such succession represent also an open
or hyphenated compound, such control function code-word having one value,
preferably the highest, when governing a solid compound and other values
when governing an exact space or hyphen position, in particular an extreme
value to make the sequence represent an open compound with a single
letter as the first element, such as 'a priori', and a value next to
such an extreme value to make the same succession of compact code-words
represent the corresponding hyphenated compound, i.e. 'a-priori', or

means for including at the end of the succession of compact code-words
a control function code-word to make such succession represent also
capitalized compounds, still preserving a desirable sorting order by
utilizing a wide spectrum of control function code-word values, using an
extreme value to govern an all-capital compound open after the first
letter, such as 'A PRIORI'.

18. A system as defined in claim 5, wherein said code conversion means
include

means for generating a compact code representation, whereby compact
code-words representing frequently occurring entity key I/O code-words,
pairs of entity key I/O code-words or larger groups of entity key I/O
code-words are made up of fewer bits and in case of less frequent
occurrences of larger number of bits.
19. A system as defined in claim 5, wherein said code conversion means include:
means for holding in store a compact code-word representation of at least one entity key attribute set, each such set representing a unique selection of entity key attributes, and
means for comparing a succession of compact code-words with the compact code-word representation of at least one entity key attribute set and for substituting one specific compact code-word for the specific part of such succession of compact code-words that is equalling the compact code-word representation of a specific entity key attribute, such specific compact code-word being drawn from a compact code being unique for the individual code-word position within the refurbished succession of compact code-words or being unique as governed by a code shift control code-word.

20. A system for storing a set of entity keys and for providing a response to a subsequent input of an individual entity key code representation, comprising:
input means for transforming part of a current entity key I/O code-word input into one digital word comprising at least one section, whereby each such section attains a current section value and when linked together a current overall section value uniquely representing such part of such I/O code-word input,
bit map memory means for holding in store bit mark representations of section values related to previously stored entity keys, and
control means connected to said input means and to said memory means for finding a current bit location using a current first section value as a relative address value, for reading out the bit value of such location to test if a previously stored entity key has produced a bit mark or not, for producing, upon a bit mark match, a section ordinal code-word from a count of the number of marked bit locations within a bit location address range below or above the address of the current bit location, for outputting an input response if no current section value remains unused, else producing a further relative address value by concatenating, in any prescribed order, the current contents of at least one remaining section with the preceding section ordinal code-word just having been produced,
for finding, within an individually section related part of the bit map, a further current bit location using such further relative address value and for repeating iteratively the functioning, section by section, as long as any current section value remains unused and the bit mark match is produced.
21. An arrangement as defined in claim 20, wherein said control means include
means for producing marked bit counts by reading out overall binary
word values from consecutive bit map sub-areas, each comprising a
succession of bit locations, and for using such overall values as entries
into a look-up table providing sub-area counts as outputs.

22. A system as defined in claim 20, wherein said memory means include
means for holding in store, at least within one section related part of
the bit map, a set of marked bit counts, each member of such set
representing a count taken within a limited address range, for providing,
related to a bit location close to the current bit location, an aggregate
count value produced from at least one such limited address range count,
and for producing the current section ordinal code-word from such
aggregate count value, adding or subtracting a marked bit count produced
for the interval between the close and the current bit location.

23. A file handling system for storing and transferring compressed
files, each file containing at least one segment, any such segment being
either a frequent segment or an infrequent segment, each segment being
input to and output from the system encoded into I/O code-words drawn
from a distinct I/O code, comprising
segment reference means for holding at least one reference segment data
set, each such reference segment data set comprising a system
representation of a special selection of frequent segments, and for
allocating a distinct RSS code-word to each reference segment, such
distinct RSS code-word being retrievable for any frequent segment, using
the frequent segment as a key, and conversely, any frequent segment being
retrievable using a distinct RSS code-word as a key,
coding means for substituting one distinct RSS code-word for each
distinct sequence of I/O code-words that is representing a distinct
frequent segment, each such distinct RSS code-word being retrieved from a
specific reference segment data set and applied inside the file handling
system as a file code-word, and for substituting a distinct sequence of
file code-words for each sequence of I/O code-words that is representing
a distinct infrequent segment, such file code words being drawn from a
distinct variable length type of code using fewer digits to represent
frequently occurring I/O code words, pairs of I/O code-words or larger
groups of I/O code-words and a larger number of digits in cases of less
frequent occurrences, and
decoding means for outputting an I/O code-word representation of a file
being retrieved from the system storage or being transferred from another
system location, thereby substituting I/O code-words for the file
code-words.

24. A file handling system as defined in claim 23, wherein said coding
means include
means for employing variable length RSS code-words to represent
frequent segments, using fewer digits to represent more frequently
occurring frequent segments and a larger number of digits in case of
less frequently occurring frequent segments.

25. A method for storing a set of entity keys and for providing a
response to a subsequent input of an individual entity key code
representation, including the steps of
transforming part of a current entity key I/O code-word input into at
least two linked digital words, which thereby each attains a current link
value and jointly a current overall link value uniquely representing such
part of such I/O code-word input,
determining a current address range for each link succeeding the first
one, using at least one current link value to find previously stored
address component values to determine the boundaries of such current
range,
comparing the current value of each link succeeding the first one with
at least one link value previously stored within such current address
range,
and producing (i) an overall link value matching condition, if for each
link succeeding the first one, the current link value matches a
previously stored link value or (ii) an overall link value non-matching
condition if the current overall link value is found to fall between
overall link values related to previously stored entity keys.
**FIG. 1B**

**INPUT SEQUENCE OF OCTET I/O CODE-WORDS REPRESENTING A PREVIOUSLY NOT STORED EK 'abb'**

<table>
<thead>
<tr>
<th>OCTET VALUES</th>
<th>LINK VALUES</th>
<th>CONTROL PROCESSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>O₀=97</td>
<td>L₁=24930</td>
<td>SYS BUS</td>
</tr>
<tr>
<td>O₁=98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O₂=98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L₂=98</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE T₁**

<table>
<thead>
<tr>
<th>B₂₀</th>
<th>B₂₁</th>
<th>B₁₀</th>
<th>B₁₀+L₁</th>
<th>B₂₂</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE T₂**

<table>
<thead>
<tr>
<th>B₂₀</th>
<th>B₂₁</th>
<th>B₁₀</th>
<th>B₁₀+L₁+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**STEPS FOR INSERTION**

1. USE L₁ TO FIND B₂₁, B₂₂
2. USE B₂₁, B₂₂ TO FIND T₂₁
3. SEARCH T₂₁ FOR L₁ VALUE
4. MOVE ALL L₂ FOR PQ>10
5. INSERT L₁ VALUE AT L₂₁
6. MAKE B₂₂=B₂₂+1 FOR P>1

**BUS OUTPUT**

EK REJECT SIGNAL

**TABLES T₂**

<table>
<thead>
<tr>
<th>B₂₀</th>
<th>B₂₁</th>
<th>B₁₀</th>
<th>B₁₀+L₁+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLES T₂**

<table>
<thead>
<tr>
<th>B₂₀</th>
<th>B₂₁</th>
<th>B₁₀</th>
<th>B₁₀+L₁+1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SYSTEM BUS**

**E1**

**F1**
INPUT SEQUENCE OF OCTET I/O CODE-WORDS
REPRESENTING THE EK 'abc'

\[ \begin{align*}
[a] &= 97 \\
[b] &= 98 \\
[c] &= 99 \\
[ab] &= 24930
\end{align*} \]

FIG. 1D

**INPUT MEANS**

\[ \begin{align*}
O_0 &= a & O_1 &= b & O_2 &= c \\
L1 &= ab & L2 &= c
\end{align*} \]

**BUS OUTPUT:**

- EK8=10
- AND/OR
- POINTER PI2
- AND/OR
- EK MATCH SIGNAL
- AND/OR
- EK REJECT SIGNAL

**SYSTEM BUS**

**CONTROL PROCESSOR**

**TABLE T1**

FIRST LINK MEMORY UNIT

**TABLES T1₂**

MAIN LINK MEMORY UNIT
ALPHIT CODE-WORD VALUES WITH CORRESPONDING ASCII I/O CODE-WORD VALUES & CODE SYMBOLS

BINARY TREE REPRESENTATION

I/O CODE SYMBOLS

I/O HEXADECIMAL NOTATION

ALPHADECIMAL NOTATION

BINARY NOTATION

NOTE 1

0^α = SYSTEM CONTROL FUNCTION
1^α = SYSTEM CONTROL FUNCTION
2^α = SYSTEM CONTROL FUNCTION
40^α = SYSTEM CONTROL FUNCTION

NOTE 2

SUB-SETS FOR MORE I/O CODE-WORDS:
4^α INITIATES A JUMP TO HIGH LEVEL,
W^α TO LOW LEVEL SUB-SET. HIGH OR
LOW GOVERNS THE SORTING ORDER.
NIBBLE CODE-WORD VALUES WITH CORRESPONDING ASCII I/O CODE-WORD VALUES & CODE SYMBOLS

BINARY TREE REPRESENTATION

FIG. 2B

I/O HEXADECIMAL NOTATION

NIBBLE HEXADECIMAL NOTATION

I/O CODE SYMBOLS

NIBBLE BINARY NOTATION

0x - 61x - a
1x - 63x - c
2x - 64x - d
3x - 65x - e
4x - 66x - f
5x - 67x - h
6x - 68x - i
7x - 66x - l
8x - 6d5x - m
9x - 6nx - n
Ax - 6jax - o
Bx - 72x - r
Cx - 73x - s
Dx - 74x - t
Ex - 75x - u
Fx - NOTE 2
Fo5x - 62x - b
F15x - 67x - g
F2x - 6ax - j
F3x - 6bx - k
F5x - 71x - q
F6x - 76x - v
F7x - 77x - w
F8x - 78x - x
F9x - 79x - y
FAx - 6ax - z
FBx - 20x - sp
FCx - 2d5x -
FDx - NOTE 1
FE5x - NOTE 1
Fx - NOTE 1
FF0x - FF5x - SUB-SET
----- - FFX - NOTE 2
FF0x - FFFFFx - SUB-SET

NOTE 1
SYSTEM INTERNAL CODE-WORDS:
FDx = SYSTEM CONTROL FUNCTION
FEx = SYSTEM CONTROL FUNCTION

NOTE 2
Fx = JUMP TO NEXT LOWER SUB-SET
FFx = JUMP TO NEXT LOWER SUB-SET
FFFx = JUMP TO NEXT LOWER SUB-SET
FIG. 2C

VARIABLE LENGTH CODE-WORD VALUES WITH CORRESPONDING ASCII I/O CODE-WORD VALUES & CODE SYMBOLS

BINARY TREE REPRESENTATION

I/O CODE SYMBOLS

I/O HEXADECIMAL NOTATION

VARIABLE LENGTH CODE-WORD
VALUES IN BINARY NOTATION

65x - e
74x - t
61x - a
6fx - o
6ex - n
72x - r
69x - i
68x - h
73x - s
64x - d
6cx -
63x - c
75x - u
6dx - m
66x - f
70x - p
79x - y
62x - b
77x - w
67x - g
76x - v
6ax - j
6bx - k
78x - x
71x - q
6ax - z

NOTE 1
FOR ALLOCATION OF OTHER
I/O CODE-WORDS AS WELL AS
SYSTEM CONTROL FUNCTIONS

ET CETERA NOTE 1
INPUT SEQUENCE OF OCTET I/O CODE-WORDS
REPRESENTING A PREVIOUSLY NOT STORED EX 'et cetera'

OCTET VALUES
00, 01, 02, 03

ALPHIT VALUES
A0, A1, A2, A3

LINK VALUES

CONTROL PROCESSOR

INPUT MEANS

FIRST LINK MEMORY UNIT

BUS OUTPUT
EK REJECT SIGNAL


B0o

NUL

SP

1 2 3

= 4

= 4

0 1 2 3

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

= 6

LOOK-UP TABLE TL0

CODE CONVERSION MEANS

FIG. 3B

OCTET VALUES REPRESENTING THE LAST PART OF THE INPUT SEQUENCE ARE NOT NECESSARY IN ORDER TO DETERMINE A UNIQUE OVERALL LINK VALUE.

L2> L2m=42 et <et cetera
L2m=42 etc
L2m=42 et etc.
L2m=42 etc.
L2m=EC etch T2m

L2< L2m=42
L2m=42
L2m=42
L2m=42
L2m=42
L2m=42
L2m=42
L2m=42

B2z

B2z+1

TABLES T2-

MAIN LINK MEMORY UNIT
INPUT SEQUENCE OF OCTET I/O CODE-WORDS REPRESENTING THE EX 'ETA'

OCTET VALUES O
- O₀, O₁, O₂, O₃

ALPHIT VALUES A
- A₀, A₁, A₂, A₃

LINK VALUES

CONTROL PROCESSOR

FIRST LINK MEMORY UNIT

INPUT MEANS

DIBIT VALUE Q
- Q = 1, i.e. 01

BUS OUTPUT
- EKS = (B₂₀ + EKS) - B₂₀ = 6

TABLES T₂ₐ

MAIN LINK MEMORY UNIT

Q = f(E₁, E₂, E₃)
- Q = 0₀  EXCEPT AS FOLLOWS:
- Q = 0₁  IF E₁ = 0ₐ AND E₃ = E₂
- Q = 1₀  IF E₁ = 0ₐ AND E₃ = 011₁
- Q = 1₁  IF E₁ = 0ₐ AND E₃ = 111₁

E₁ = 000₀  E₂ = 000₁  E₃ = 000₁

TABLE TL₀

CODE CONVERSION MEANS (PART OF INPUT MEANS)
INPUT SEQUENCE OF OCTET I/O CODE-WORDS
REPRESENTING THE EK 'et cetera'

FIG. 5B

INPUT MEANS

CODE
CONVERSION
MEANS

L1=4EC
L2=E9
L3 L4 L5 L6 L7 L8

CONTROL
PROCESSOR

FIRST LINK
MEMORY UNIT

OUTPUT
MATCH
AND/OR
EKS=9

T3S-T8S
EKS=0001\(^x\),001\(^x\)
L3

B3\(_o+0001\(^x\)
001\(^x\),P\(_m\)

T3S

B3\(_o\)

B4\(_o+0000\(^x\)
001\(^x\),9\(_m\)

T4S

B4\(_o\)

B4\(_o+0000\(^x\)
001\(^x\),9\(_m\)

T5S

B5\(_o+0000\(^x\)
001\(^x\),M\(_m\)

T6S

B6\(_o+0000\(^x\)
001\(^x\),5\(_m\)

T7S

B7\(_o\)

B7\(_o+0000\(^x\)
001\(^x\),0\(_m\)

T8S

TABLES
T2\(_o\)

MAIN LINK MEMORY UNIT

ADDITIONAL LINK
MEMORY UNIT

1) EK MATCH IF
L9 DOES NOT EXIST

T7S=0000\(^x\),000\(^m\)

B8\(_o+0000\(^x\)
000\(^m\),2\(_m\)

T8S

B9\(_o\)

B9\(_o+0000\(^x\)

### INPUT SEQUENCES TO LONG TO BE ACCOMODATED WITHIN $A_0$ THROUGH $A_9$, LIKE THE CURRENT ONE 'ETCETERA', WILL CAUSE CODE CONVERSION MEANS 46 TO CHANGE THE VALUE IN THE BIT POSITION SUCCEEDING ALPHIT $A_9$, I.E. BOX 59, FROM 0 TO 1 AND THE PRECEDING 15 BIT POSITIONS WILL NOT BE USED TO ACCOMODATE ALPHIT VALUES BUT A SEQUENCE NUMBER IDENTIFYING AN ADDRESS IN THE ADDITIONAL LINK LOOK-UP TABLE WERE A REPRESENTATION OF THE COMPLETE TAIL IS STORED.
INPUT SEQUENCE OF OCTET I/O CODE-WORDS
REPRESENTING THE EK 'etaonrih'

CODE CONVERSION MEANS

FIRST LINK MEMORY UNIT

CONTROL PROCESSOR

OUTPUT

TABLES T2a & T3a
INTERMEDIATE LINK MEMORY UNIT

TABLES T4pa
MAIN LINK MEMORY UNIT

FIG. 6C
**FIG. 6D**

**INPUT SEQUENCE OF OCTET I/O CODE-WORDS**

**REPRESENTING THE EK 'et cetera'**

<table>
<thead>
<tr>
<th>CODE CONVERSION MEANS</th>
<th>INPUT MEANS</th>
<th>34</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ao A1 A2 A3 A4 A5 A6 A7</td>
<td>S1 S2 L1 L2 L3 L4 L5 ... Lz</td>
<td></td>
</tr>
</tbody>
</table>

**CONTROL PROCESSOR**

**FIRST LINK MEMORY UNIT**

**INTERMEDIATE LINK MEMORY UNIT**

**MAIN LINK M.U.**

**ADDITIONAL LINK MEMORY UNIT**

**OUTPUT**

\[ P_{par} \]

AND/OR EKS

**TABLE T1**

**TABLE T2**

**TABLE TM**

**TABLE TM2**

**BIT MAP MEMORY MEANS (PART OF CODE CONVERSION MEANS)**

**SUBSTITUTE SHEET**

**ISA/SE**
**FIG. 6E**

**INPUT SEQUENCE OF OCTET I/O CODE WORDS**

<table>
<thead>
<tr>
<th>Ao</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
<th>A10</th>
<th>A11</th>
<th>A12</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1'</td>
<td>L2'</td>
<td>L3'</td>
<td>L4'</td>
<td>L1''</td>
<td>L2''</td>
<td>L3''</td>
<td>L4''</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIRST INPUT MEANS**

1. **FIRST LINK MEMORY UNIT**
2. **INTERMEDIATE LINK MEMORY UNIT**
3. **MAIN LINK MEMORY UNIT**

**SECOND INPUT MEANS**

1. **FIRST LINK MEMORY UNIT**
2. **INTERMEDIATE LINK MEMORY UNIT**
3. **MAIN LINK MEMORY UNIT**

**EXS'**

<table>
<thead>
<tr>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
</tr>
</thead>
</table>

**EXS''**

**FIRST INPUT MEANS**

1. **FIRST LINK MEMORY UNIT**
2. **INTERMEDIATE LINK MEMORY UNIT**
3. **MAIN LINK MEMORY UNIT**

**SECOND INPUT MEANS**

1. **FIRST LINK MEMORY UNIT**
2. **INTERMEDIATE LINK MEMORY UNIT**
3. **MAIN LINK MEMORY UNIT**

**OUTPUT**

- \( P_{par} = f(B_{par}, B_{par+}, L4) \)
- \( B_{par} = B_{par0} + f(B_{2p}, B_{2p+}, L2, L3) \)
- \( B_{par+} = B_{par0} + f'(B_{2p}, B_{2p+}, L2, L3) \)

**INTERMEDIATE LINK MEMORY UNIT**

**MAIN LINK MEMORY UNIT**
FIG. 7B

INPUT SEQUENCE OF OCTET I/O CODE-WORDS
REPRESENTING THE EK 'etcetera'

CODE
CONVERSION
MEANS

CONTROL
PROCESSOR

INPUT MEANS

FIRST
LINK
M.U.

INTERMEDIATE
LINK
MEMORY UNIT

MAIN
LINK
M.U.

ADDITIONAL
LINK
MEMORY UNIT

OUTPUT

Pseud
AND/OR EKS

[etcetera] 00...0101^B
L1-LZ= [etcetera], X, N0-Nz
X=11^B
N0=0^x
Nz=F^x
N4=
N5=

PREFIX
ATTRIBUTE SET
LOOK-UP TABLE

SUFFIX
ATTRIBUTE SET
LOOK-UP TABLE

N1=0^x
U=0^x

E=0^x
N0=0^x

10
32,34
58 14
50
70
[etcetera]
FIG. 8A

VARIABLE LENGTH RSS CODE-WORDS
REPRESENTING FILE SEGMENTS,
BINARY TREE REPRESENTATION

RSS = 10^x - 5F^x, I.E.
80 RSS CODE-WORDS,
REPRESENTING 80 MOST
FREQUENT FILE SEGMENTS

RSS = 600^x - AFF^x, I.E.
1280 RSS CODE-WORDS
= 1280 FILE SEGMENTS

RSS = B000^x - FFFF^x, I.E.
20480 RSS CODE-WORDS
REPRESENTING 20480
FILE SEGMENTS
# INTERNATIONAL SEARCH REPORT

**International Application No:** PCT/SE87/00406

## I. CLASSIFICATION OF SUBJECT MATTER

According to International Patent Classification (IPC) or to both National Classification and IPC:

G 06 F 5/00, H 03 M 7/30

## II. FIELDS SEARCHED

<table>
<thead>
<tr>
<th>Classification System</th>
<th>Classification Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPC 4</td>
<td>G 06 F 5/00, 12/00, /02; H 03 M 7/30-/50</td>
</tr>
<tr>
<td>IPC 3</td>
<td>H 03 K 13/22</td>
</tr>
<tr>
<td>US Cl</td>
<td>235:310; 340:347DD; 364:200, 900</td>
</tr>
</tbody>
</table>

*Documentation Search other than Minimum Documentation to the extent that such Documents are Included in the Fields Searched

SE, NO, DK, FI classes as above

## III. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of Document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to Claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>US, A, 4 586 027 (TSUKIYAMA ET AL) 29 April 1986</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US, A, 4 560 976 (FINN S G) 24 December 1985</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US, A, 4 558 302 (WELCH T A) 10 December 1985</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US, A, 4 491 934 (HEINZ K E) 1 January 1985</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US, A, 4 428 063 (FOURCADE P) 24 January 1984</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US, A, 3 821 711 (ELAM R R ET AL) 28 June 1974</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>WO, A1, 86/00479 (TELEBYTE CORPORATION) 16 January 1986</td>
<td></td>
</tr>
</tbody>
</table>

* Special categories of cited documents:

- **A** document defining the general state of the art which is not considered to be of particular relevance
- **E** earlier document but published on or after the international filing date
- **L** document which may throw doubt on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- **O** document referring to an oral disclosure, use, exhibition or other means
- **P** document published prior to the international filing date but later than the priority date claimed

**T** later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

**X** document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

**Y** document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

**Z** document member of the same patent family

## IV. CERTIFICATION

**Date of the Actual Completion of the International Search:** 1987-11-19

**Date of Mailing of this International Search Report:** 1987-11-26

**International Searching Authority:** Swedish Patent Office

**Signature of Authorized Officer:**

[Signature]

Form PCT/ISA/210 (second sheet) (January 1985)
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of Document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to Claim No</th>
</tr>
</thead>
<tbody>
<tr>
<td>P,A</td>
<td>GB, A, 2 172 127 (FERRANTI PLC) 10 September 1986</td>
<td></td>
</tr>
</tbody>
</table>

Form PCT/ISA/210 (extra sheet) (January 1985)