한글 한자 자동변환이 가능한 WC

83307

이 기 식 서 용 무
한국과학기술원 소프트웨어 개발실

WORD-PROCESSING SYSTEM CAPABLE OF AUTOMATIC TRANSFORMATION FROM HANGUEL TO CHINESE.

KISIK, LEE — YONGMOO, SUH
K A I S T — SOFTWARE DEVELOPMENT CENTER

ABSTRACT

This paper describes the result of a series of fundamental researches on a word processing system, named WORD80.

Given the text consisting of Hanguel characters, Chinese characters, English alphabets and other special characters, it can provide ready-to-see printout, after having processed according to the commands used to control the output format of the text. Since each Hanguel character in input text is made up of and is represented as some components (vowels and consonants), it uses the deterministic finite automata for the formation of Hanguel from its components. When developed at first, the WORD80 can perform Hanguel to Chinese transition character by character. But it was thought to be more practical to do this word by word.

To cope with this situation, we investigated about 600,000 Hanguel words whose roots stem from Chinese words. We selected about 40,000words out of these based on their frequencies, and made them into a dictionary which will be used for the automatic transition from Korean to Chinese words, after making various experiments on the desirable structure of the dictionary. As is often the case, minimum space and minimum access time was adopted as the criteria for choosing the structure.

1. INTRODUCTION

Of late, quite a few industries of the world have showed deep interest in the Office Automation. It is a well-known fact that the computer processing of the natural languages came into being as one of the important branches in the computer application field some years ago.

Seeing that among various aspects of Office Automation word processing plays a great role, we developed a word processing system, say WORD80, which

suits the processing of Korean text,

Because Koreans commonly use Chinese Characters as well as Korean characters, we updated the system so that its input can be composed of Korean, Chinese characters and others.

Now the WORD80 has a new function of automatic transition from Hanguel to Chinese words, using a dictionary of about 40,000 words,

Building a dictionary is a first step toward the natural language translation.

Some research groups in Korea are positively studying of word-processing. This paper depicts the configuration of WORD80 , the algorithm of Hanguel formation from its components, format [illegible] dictionary file.

Of course, this WORD80 system have something to be desired. Suggestions on this are described at the end.

2. WORD80 SYSTEM CONFIGURATION

2.1 HARDWARE OF WORD80 SYSTEM

Our word-processing system WORD80 is made up of three main parts, CPU, input device and output device, as is depicted in figure 2.1.

It is estimated that 8-bit micro processor is appropriate for the word processing, in that the input data to word-processing system is usually of 8-bits. So we developed WORD80 using CROMEMCO micro computer SYSTEM THREE the CPU of which is a 8-bit microprocessor, Z80. Diskette controller is attached to the CPU so that it can perform the file management in diskettes, one of which is a dictionary diskette used for the automatic Hanguel-to - Chinese transition.

We are using CRT terminal made by OSI co. of Korea as our input device. But other CRT's compatible to this can also be used for inputing data. Its general characteristics are as follows:

- o size of the screen: 12 inches in diagonal
- o screen capacity: 64 character/line X 16 line
- o transmission speed: 110 to 9600 baud
- o the size of each character: 11 X 13 dots for Hanguel
  7 X 9 dots for English alphabet.

Trilog 100 which is used as the output device is a line matrix/plotter. Its printing speed is 250 LPM in print mode or 15 inches per minute in plot mode. It has a line buffer of 132 characters and has mechanically 44 hammers with each taking care of 3-character width. The diameter of each hammer is 0.020 inch, which restricts the number of dots for each printed character.

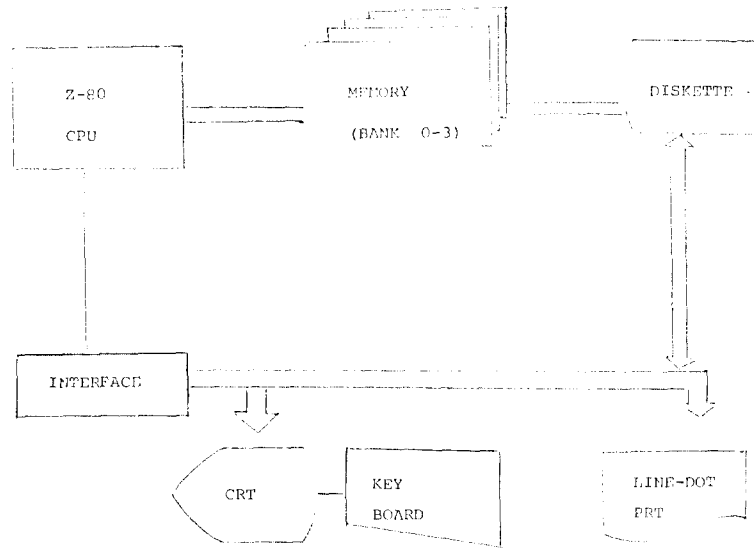


Figure 2.1 SYSTEM CONFIGURATION DIAGRAM.

2.2 SOFTWARE OF WORD80

For the word-processing, we generated the font images of about 3,000 Chinese characters, 145 Hanguel characters, 52 alphabets of upper and lower cases, some special characters and about 40 numeric characters. These font images requires, of course, a large amount of the main memory. In most cases, 14 X 16 to 22 X 20 dots for each Hangel, 24 X 24 or more dots for Chinese character and 7 X 9 dots for each English alphabet are needed in order for these characters to look good when printed.

The bigger the character is, the more storage it requires. For this reason and due to the big diameter of the printing hammer, each Hanguel and each Chinese character is of 18 X 18 dots and the other of 12 X 16 dots. These include 2 bits of gaps between lines and characters.

Computed in this way, the total sum of storage for the font images of all characters and for the word prossing program is 160 K-bytes, which is beyond the size of the address space of Z-80 CPU. This forced us to use the the concept of memory banking. The self-explaining memory map is represented in Figure 2.2.

WORD80 consists of one main and 16 subprograms and the overall structure and relation can be depicted as in Figure 2.3. Here we list the various functions of WORD80 system.

o tab setting

o underlining

o centering of line

o printing in AS-is mode

o capitalizing

o capitalizing of the first character of each word

o forcing a new paragraph

o right justification

o double/single spacing

o etc.

o indentation

o line feed

o page ejecting

o elongating vertically and/ or horizontally

o title printing on every page

o page numbering

o automatic transition from Hanguel to Chinese word by word

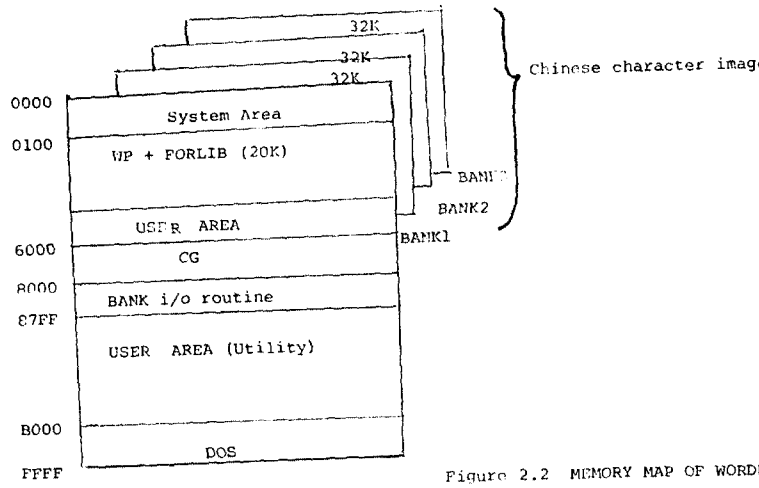


Figure 2.2 MEMORY MAP OF WORD80 SYSTEM

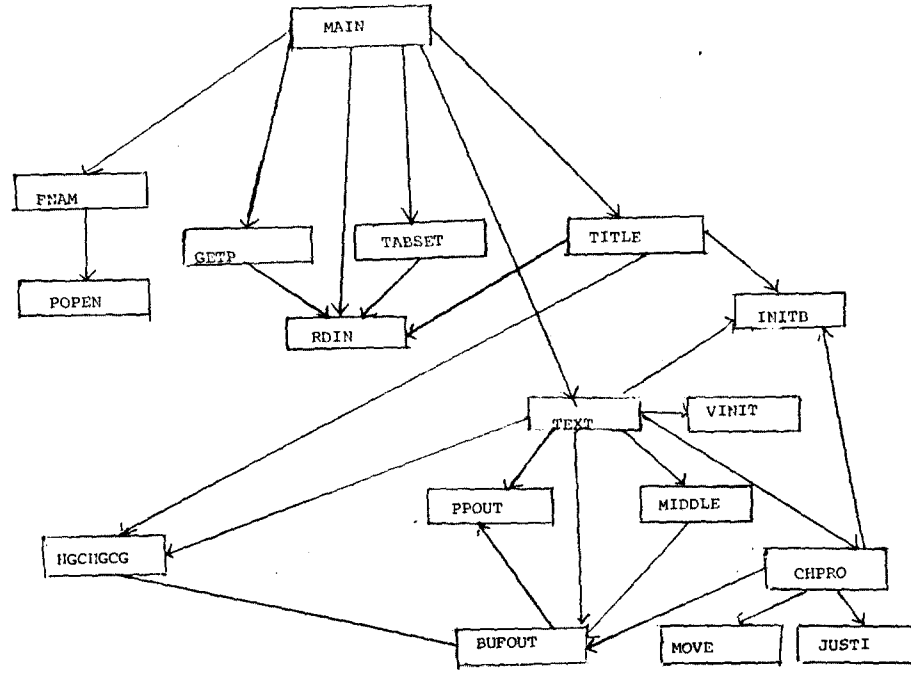


Figure 2.3 PROGRAM OF WORD80

3. WORD-PROCESSING ALGORITHM

There are 14 consonants and 10 vowels in the set of preliminary elements in Korean language system and the combination of these components forms one Hanguel character. The number of the combinations possible is so great that we can't input Hanguel data by character. Therfore WORD80 system had to adopt the method of inputting by element. The number of elements used in our system is is 33, of which 19 are consonants and 14 are vowels.

These elements are on the key board of the input device. This input method implies that we need a principle to form Hanguel characters from its elements given in the input text.

For the Hanguel formation and the command processing, the theory of formal languages was used. That is, all the input characters are scanned according to a transition diagram or a Deterministic Finite Automata (DFA).

Following the notation of Aho & Ullman, we can represent the DFA used for laying out WORD80 system like this:

DFA $M=(Q, \Sigma, \delta, q_0, F)$

$Q=\{S, E, Qi \mid i=1 \ldots 19\}$

$\Sigma=\{$), c, v, h, β, α, IC, ID, IE, b$\}$ where

): symbol going before command character

Q: Set of state in the transition diagram

Σ: set of input symbols

C: 19 consonants of Hanguel such as

ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅅ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ, ㄲ, ㄸ, ㅃ, ㅆ, ㅉ.

v: 9 vertical vowels of Hanguel such as

ㅏ, ㅑ, ㅓ, ㅕ, ㅐ, ㅒ, ㅔ, ㅖ, ㅣ

h: 5 horizontal vowels of Hanguel such as ㅗ, ㅜ, ㅠ, ㅡ.

b: blank

β: one of !,? and .

IC, ID, IE: chinese character mode, Hanguel mode, English alphabet mode symbols respectively.

α: other input symbols

q_0: initial state (S)

F: set of final states, (Q1, Q18, Q19)

δ: mapping from $Q \times \Sigma$ to Q

E: error state.

This DFA can be drawn into two state transition diagrams, Figure 3.1 and Figure 3.2

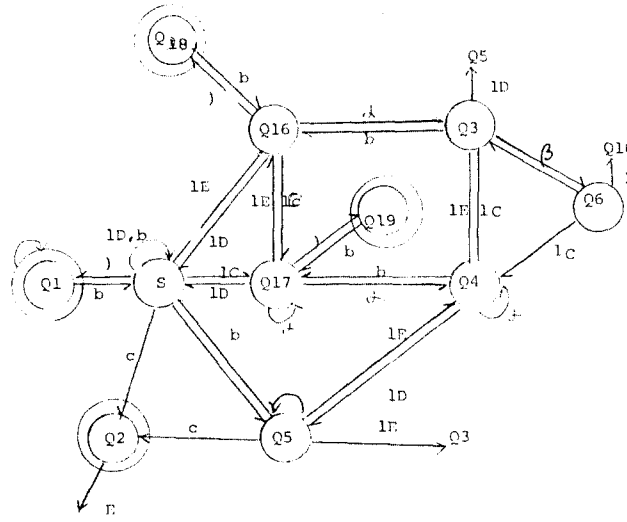


Figure 3.1

state transition diagram for the overall input processing.

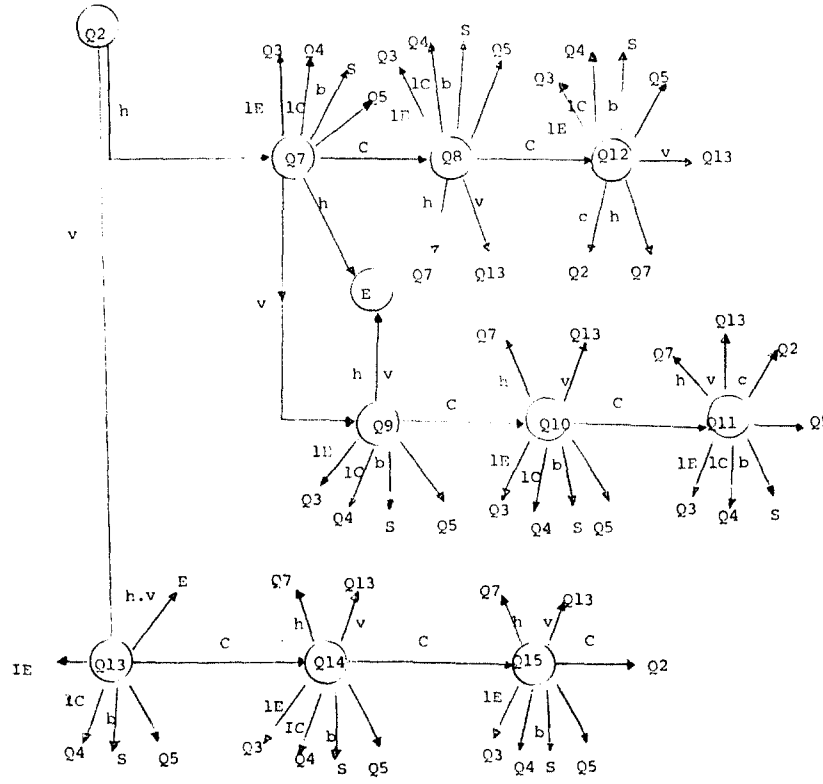


Figure 3.2. State transition diagram of Finite Automata

4. AUTOMATIC KOREAN-TO-CHINESE TRANSITION.

4.1 Collection of words and data structure for a dictionary.

As the need for natural language translation grows, it is absolutely required to build a dictionary into a computer system. To be more valid, it must contain as many words as possible, and as many attributes as conceivable. But in our case, we had the codes corresponding to each Chinese-root word contained in the dictionary, for this will be used only for the transformation.

In order for WORD80 to carry out the automatic transition from Hanguel to Chinese-root words among the ones commonly used in daily life, economy, science, law and literature, etc. The total number of words collected is 667,322, of which only 43,507 words of high frequency became candidates for a dictionary.

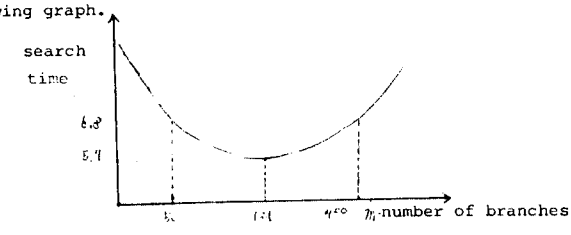
When building a dictionary, we have to make it as small and as fast as it can be. Taking this into consideration, we thought the sequential file was not so good as the indexed file, because the amount of data collected was too great. Among the commonly-used indexed files, Hash and multi-way tree files are the most well-known ones. Therefore we made various experiments on Hash and B-tree, one of the multi-way tree, and then determined the B-tree as the data structure of our dictionary. In the experiment, we made both files of each word-length.

4.2 B-tree file as a dictionary.

B-tree is one of the multi-way search tree. If the branches of a multi-way tree are connected in one direction, it may be as good as a sequential file. To avoid this possibility, we come to think of the balancing of a multiway tree.

Thus made is a B-tree,i.e., balanced multiway tree. The property of being balanced in a B-tree is due to the fact that its root is generated anew when necessary, while in a multiway tree, its root node is fixed all along.

It is important to choose the proper number, m of branches of a B-tree. Theoretically, the best choice of it can be explained by the following graph.



The value of m in the above graph is determined by the factors such as seek time, latency time and character transmission time, etc. But actually the value of m is determined by the input buffer size, 128 bytes in the case of CROMEMCO micro computer system.

B-tree file is composed of a index part and a file part. Hanguel codes are forming the index part(B-tree) and chinese codes are forming a real file. Each node of a m-way B-tree is of the type n,Ao,(K1,A1,B1), (K2,A2,B2),....,(Kn,An,Bn), where $Ai, 0 \leq i \leq n$ are pointers to the subtrees, Ki, $1 \leq i \leq n$ are the key values, i.e., Hanguel codes, and Bi, $1 \leq i \leq n$ are the pointers to the corresponding entry of a file part. (n: # of key values in a node). The files of chinese codes contain link fields for the processing of synonyms.

The program that builds a dictionary from the input data, initializes the pointers to the files and to the index tables, and then stores them in block Ø of a dictionary diskette.

After reading the Hanguel and Chinese codes and its length from the CRT, it searches the index table for the node which contains the input key (Hanguel code) and then searches the node for the input key, following the binary search algorithm. It there exists the input key and there is no input Chinese code in the file, the Chinese code is written in the file.

The Hanguel-to-Chinese word transition is done by calling a subroutine using a command of a word processor, WORD80.

4.3 Hash file as a dictionary.

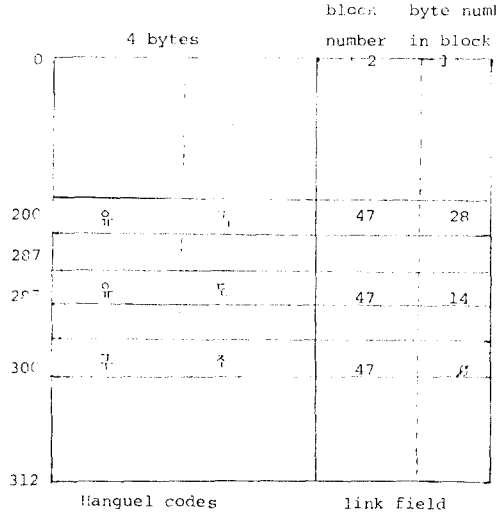
Hash techingue is commonly used by most compilers when searching symbol tables. When it comes to telling of Hash, to determine Hash function and the method to solve the collision is the most important thing to think of. There are many kinds of Hash functions: division, middle square, folding etc. Also, there are many collision-solving techniques to each Hash function: chaining, rehashing, open addressing. Since it is known that as a rule Hash function by division with chaining is the most desirable, we also adopted this when building a dictionary.

Given the input key (Hanguel code) the hash function devides it by 2 bytes and after adding these, divides this sum by the prime number.

The remainder is used as an index to the Hanguel code table. If collision occurrs, it checks whether the input key is synonym or not. In the case that synonym causes the collision to occur, the chaining is used, and if this is not the case, add-the-hash rehashing is used as a collision-sol-

ving method. If you are to reduce the frequency of collision, you have to allocate enough memory as an index table so that the loading factor can be small.

Let's assume that the hash values of '[illegible]' and '[illegible]' are 300,300 and 200 respectively and the prime number used by the hash functionis 313. Then the input data 構造, 救助, 柔道, 儒道, 有機, 有期, 誘導 makes hash file of Figure 4.1.



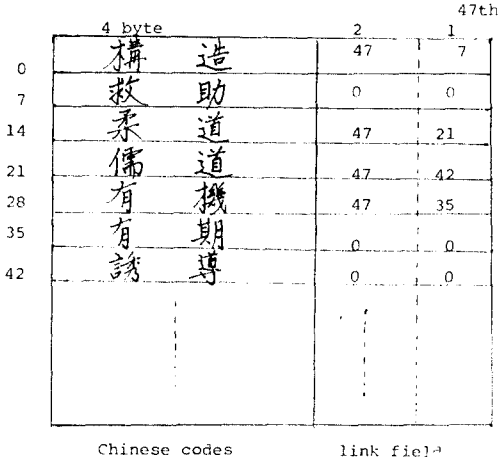


Figure 4.1 example of a Hash file

4.4 Comparison of Hash and B-tree.

We have described the data structures desirable for a dictionary. Now we are at a point of comparing these two structures in terms of speed and space, which are the usual criteria for comparing the softwares. Both the service programs using B-tree program requires 20K bytes and Hash program repuires 9.5K bytes. The ratio of the file-accessing times of these service programs is thought to be the same as that of the times spent in building that file. This ratio is represented in table 4.1. And the space comparison in table 4.2.

table 4.1 Comparison of file access time

number of word	B-tree(sec)	Hash(sec)
100	127	117
200	242	229
300	377	350
400	492	483
500	602	618
600	707	753
700	807	903
800	905	1050
900	1000	1198
1000	1094	1347

table 4.2 Space Comparison

word length	index table				file of Chinese character codes
	B-tree		Hash		
	A	B	C	D	
1	92	183	98	123	59
2	1,641	801	1,969	2,188	1,094
3	1,273	3,111	1,590	1,818	909
4	929	1,563	1,257	1,485	676
5	222	313	310	387	172
6	88	171	105	131	66
7	27	43	34	45	20
subtotal	4,272	6,185	5,343	6,177	2,996
TOTAL	7,268	9,181	8,339	9,173	

A: the case we assume each node is full

B: we take minimum number of nodes at the maximum number of level

C: the loading factor is 0.6

D: the loading factor is 0.5

As might be expected, the dictionary using B-tree concept had the advantage over the dictionary using the Hash concept, which is reflected in the foregoing tables. This result drove us to build a dictionary based on the B-tree algorithm.

5. CONCLUSION

So far, we have discussed the word-processing system with the capability to process Hanguel, Chinese as well as English characters and the data stucture for a dictionary which will be used when transforming Korean words to Chinese ones. But our system WORD80 as it stands, has something to be desired more.

First, the input buffer of the CPU is so small that we can't select the appropriate m value (the number of ways of a B-tree).
Sooner or later, the CPU of our system will be replaced with the 16-bit CPU.

Secondly, the input device, preparing the input text to WORD80, also has to be replaced by some other device with many valuable functions.

Finally, it needs a better for the screen editing facility to be included in the WORD80 system.

Though we selected about 40,000 words out of about 660,000 words collected from various fields and built them into a dictionaly, to make it what it should be. We have to study on the languages themselves intensively.