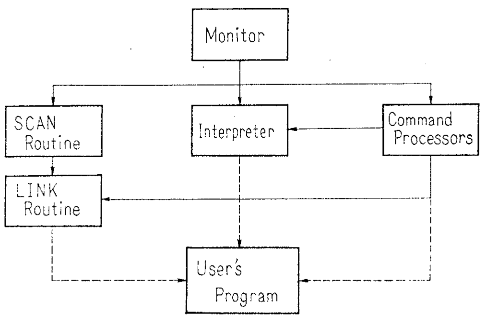
Conversational FORTRAN : KEIO System

Norihisa Doi*, Ken'ichi Harada* and Masakazu Nakanishi**

Introduction

The KEIO (Keio Elementary Instructive Operating) system is a conversational FORTRAN processor and has been implemented for the KEIO-TOSBAC Time-Sharing System at Keio University. The purpose of this system development is experimental research for software system with man-machine interaction. The followings are the functions required on the system; immediate error diagnoses, easy correcting and modifying without recompilation, debugging aids at source language level, accessibility of user program information, etc. On the implementation, we are imposed such restrictions as a shortage of small size main memory, a lack of random access mass storage and no dynamic relocation hardware. Furthermore, the user is allowed to use only 4K words drum area, in which all information of the user program must be hold. Considering these situation, we adopted the incremental compilation and interpretive-execution technique into the processor. The configuration and the memory layout of KEIO system are shown in Fig. 1, and Fig. 2.



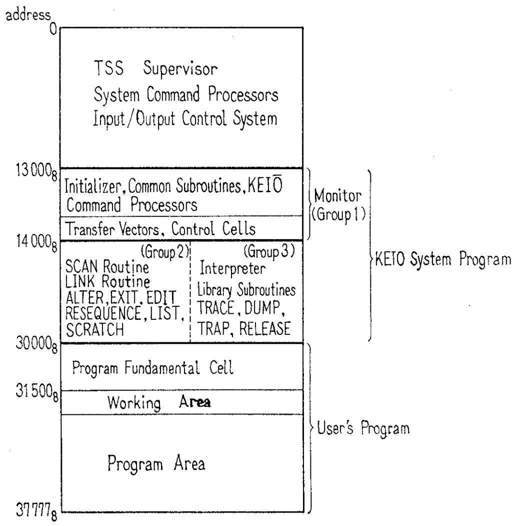


Fig. 1. The configuration of the KEIO system. Fig. 2. The memory layout of the KEIO system.

This paper first appeated in Japanese in Joho-Shori (the Journal of the Information Processing Society of Japan), Vol. 10, No. 2 (1969), pp. 68-77.

* Keio Institute of Information Sciences.

** Department of Administraction Engineering, Faculty of Engineering, Keio University.

The grammar of KEIO

The grammar of KEIO bases on FORTRAN II. An available I/O device is terminals (electric typewriter) only, for users. So, some modifications are given for the language. For example, data are read from terminals by INPUT statement.

READ statement is used for reading data from list of data which were given by DATA statements.

CRLF statement appoints the number of times of carriage return.

Some commands are ready for altering or tracing a program and taking an information about it. The commands which have to appoint a position of statement use line numbers which are assigned to each statement.

Internal Structure of User's Program

The statement which is read is checked grammartically by SCAN routine. As using some techniques in SCAN, an inefficiency of the executional speed which is caused by using a interpreter is saved. But, on the other hand, some cells are generated after arranging elements. This technique lighten a burden to minimum for a interpreter routine in execution time.

For a user's program an internal language which is composed by the cells is only remained and a source image of the program is broken. And when user wishes a program list, the source image regenerated from this internal language is given.

Now, there are two kinds of cells; the element cell and the statement cell. The element-cell is composed of the information about each element. The statement-cell is generated by decomposing a statement to its elements and replacing it to available order by SCAN routine. One statement-cell is generated for one statement.

Element-Cell

Element-Cell (EC) consists of 8 words. For a element begun with an alphabet, 26 chaines are made in alphabetical order. Also, for real constant, integer constant and statement number, each chain is made.

Statement-Cell

Statement-Cell (SC) is a block of variable length. The first 5 words are fixed part which given the information about a kind of statement, a line number and a sequence. Statement-Cell is linked in order of a line number. Generally, the elements of statement are given by index-address of EC when it is variable, constant, function name, or statement number. And when it is operator, they are given by the address of operator-table in KEIO monitor (Fig. 3).

An arithmetic statement is held by the reverse polish notation.

FORTRAN statements and comments are put in the variable part as source image. The constants enumerated in DATA statements are translated to the internal expression and put in the variable part.

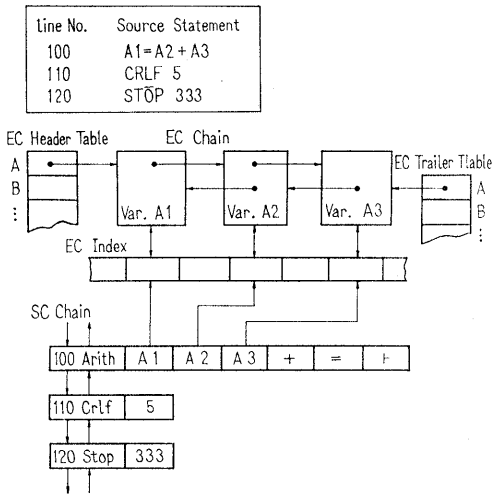


Fig. 3. The interrelation of SC and EC.

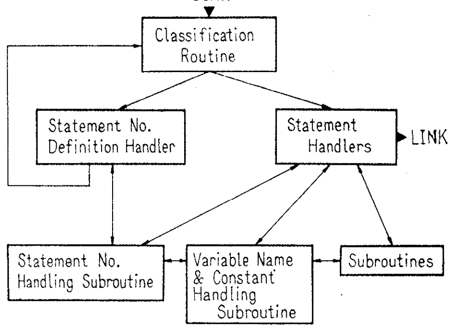
SCAN Routine

The faculty of SCAN Routine is decomposing the user's statement to SC and EC, with checking grammartically. It is not concerned with operating mode. In this case, the generated cells are not linked to the program area directly, and are composed in the working area in order. The outline of SCAN Routine is shown in Fig. 4.

Subroutine handling the variable names and constant is the largest routine in SCAN Routine.

LINK Routine

The faculty of LINK Routine is to chain SC and EC generated by SCAN routine, in order. The general flow of LINK is shown in Fig. 5.


SCAN
Classification Routine
Statement No. Definition Handler
Statement Handlers
LINK
Statement No. Handling Subroutine
Variable Name & Constant Handling Subroutine
Subroutines

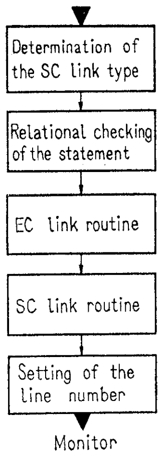

Determination of the SC link type
Relational checking of the statement
EC link routine
SC link routine
Setting of the line number
Monitor

Fig. 4. The block diagram of SCAN routine. Fig. 5. The block diagram of LINK routine.

Interpretive Routine

Interpretive Routine interprets SC chain in the program area by the medium of EC, namely, this routine executes the user's program actually. The general flow of Interpretive Routine is shown in Fig. 6.

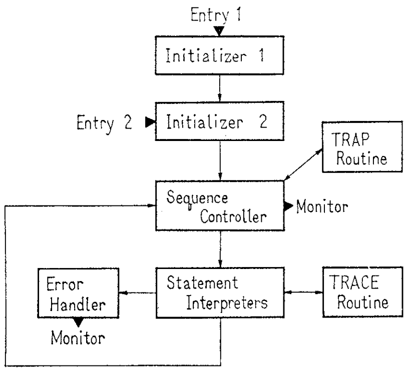


Fig. 6. The block diagram of INTERPRETER.

Conclusion

In development of the KEIO system, we encountered two kind of problems as follows.

1. The problem based on the bohavior of T. S. S.
2. The problem based on the language specification of FORTRAN

The first problem is that the execution may be interrupted by the turnning on the interruption switch, for conversation with a terminal. When user's program execution is interrupted before the end of the interpretation of a statement, the status of switches and the value of the registers may be incompletely suspended. We adopted the logical interruption mechanism to solve the problem. When supervisor detects the interruption from a terminal, it sets the flag to the specific cell in subsystem's communication area. As the Interpretive Routine scans the cell at the end of interpretation of each statement, the routine is able to detect the condition with suitable timing to itself.

The example for the second problem is that large amount of the processing time has to be spend for an alteration of DO range, DIMENSION statement. For this problem, some restrictions have been added on the FORTRAN grammar.

Acknowledgment

Thanks to Professor Shoji Ura for his advices, and to Y. Ohno, H. Ishiwata, and M. Matsumoto for their hard workings.