Software for a small research computer

Kerr, Jan

Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 1967-10

ERB-776

UNCLASSIFIED

NATIONAL RESEARCH COUNCIL OF CANADA

RADIO AND ELECTRICAL ENGINEERING DIVISION

SOFTWARE FOR A SMALL RESEARCH COMPUTER

IAN KERR

OTTAWA

OCTOBER 1967

ABSTRACT

This paper is written to describe a software system for a small digital computer used for research purposes.

CONTENTS

FIGURES

SOFTWARE FOR A SMALL RESEARCH COMPUTER

- Ian Kerr -

INTRODUCTION

The role of computers in today's complex world is one of ever increasing importance. Computers are now being assigned to tasks which a few years ago they were incapable of performing. These advances are the fruits of research into ways of improving both the computers themselves, and the programs which direct their activities.

The Data Systems Section of the Radio and Electrical Engineering Division of the National Research Council is conducting research into both hardware and software aspects of digital computers. The work is directed towards a better understanding of systems aspects of small computers, with particular application to scientific research. One area which is receiving particular attention is that associated with man-machine communications problems.

This paper describes some of the systems software which has been developed specifically to aid man-machine interaction, both during the development of programs and during their operation and evaluation.

HARDWARE SYSTEM COMPONENTS

The hardware currently available to the user includes the following items:

1. A general purpose digital computer.

2. Digital magnetic tape recorders. These units can be used either for input/output functions, or as external storage devices for the temporary storage of programs or intermediate results.

3. Teletype machines. These units can either prepare printed page or punched paper tape output, or provide keyboard or punched paper tape input. The units are slow, operating at about 10 characters per second.

4. High speed paper tape punch and reader. This unit can prepare punched paper tape output at the rate of 110 characters per second, or read punched paper tape at 300 characters per second.

5. Cathode-ray tube display unit and light pen. Under program control, this unit can display on what resembles a 21-inch television picture tube, characters,

straight and curved line segments, and points to make up any desired picture. The light pen is a photo-sensitive device which resembles a pen. When pointed at a displayed picture element it interrupts the computer allowing the program to determine at which picture element the human operator was pointing the light pen.

6. Auxiliary keyboard. This keyboard, situated close to the display, can be used used in conjunction with the display as an input to the computer.

7. Analog-to-digital and digital-to-analog converters. These units allow the computer to measure analog quantities and to produce analog signals. They are useful for controlling experiments by computer as the experiment is in progress.

SOFTWARE SYSTEM COMPONENTS

The software system used is shown in Fig. 1. Its aim is to facilitate the development of a working program. Anyone who has tried his hand at programming will appreciate how difficult and time-consuming a task it can be to remove all the logical errors from a program, or to modify a program already working. Speeding up this process allows the research oriented programmer to be more concerned with the concepts and implications of his program than with debugging it. In the diagram, the blocks indicate system programs, the solid lines indicate data flow within the memory and the dashed lines indicate data flow external to the computer (usually on paper tape).

Input-Output System

The actual programming required to read from or write on a peripheral device depends to a large extent upon which device it is. For example, magnetic tape units must read or write complete blocks of characters at a time; they cannot deal with single characters. Paper tape readers and punches, on the other hand, read or punch single characters. To keep programs independent of devices and to avoid unnecessary duplication, the Input-Output system handles input and output functions for most programs. It consists of a handler for each device and an I/O driver.

Any program wishing to read from or write on a peripheral device transfers control to the I/O driver and supplies it with the device number to use, the address of where the output data are stored or where the input data are to be stored, how many words are to be transferred, and whether an input or output function is desired. The driver then decides which handler to use, supplies it with most of the same information, and then branches to it. The handler is responsible for issuing the necessary commands to the device to cause it to

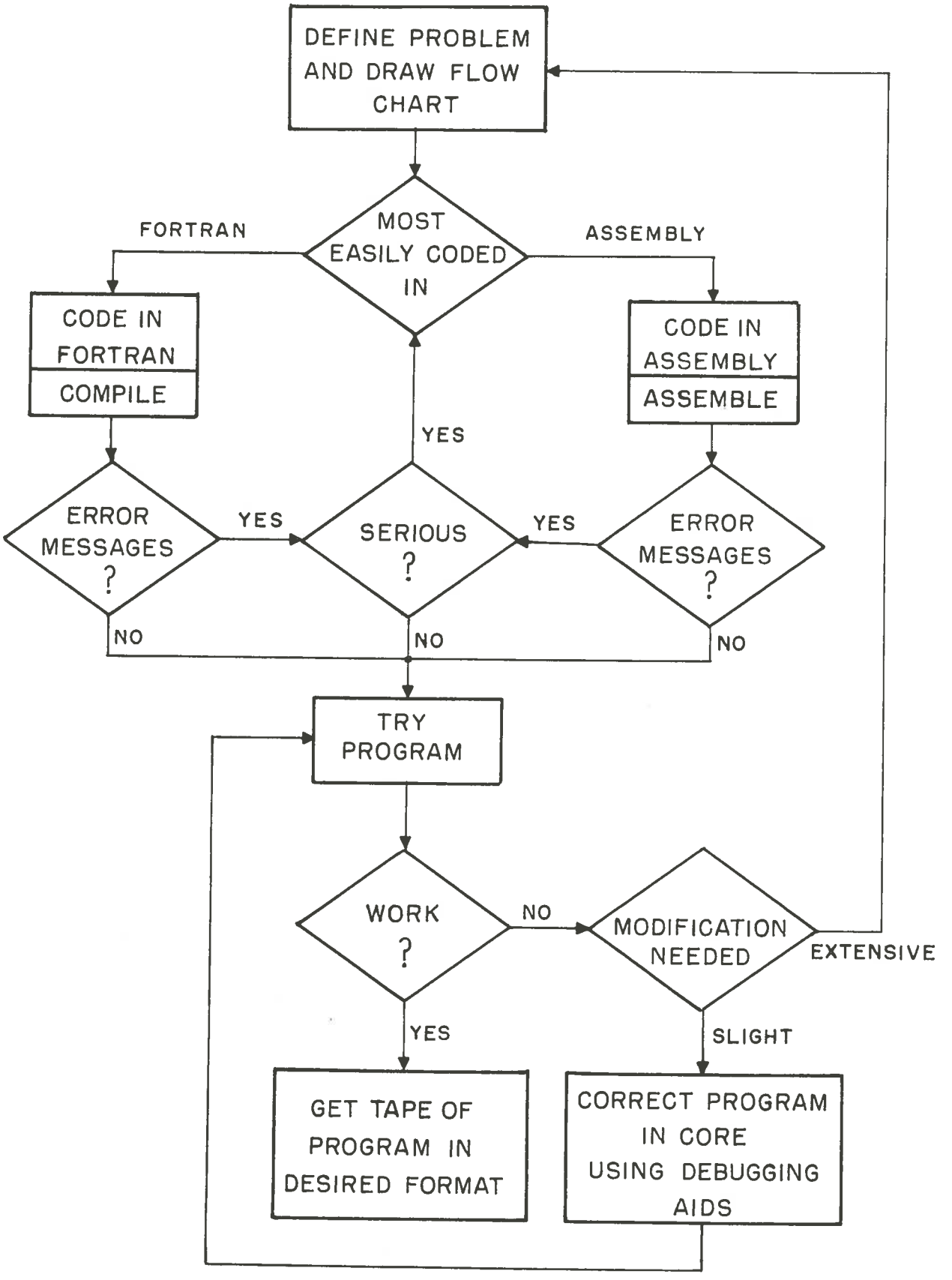

Fig. 1 Software organization

operate, to effect the data transfer, and to warn the operator of any malfunction of the device. When finished, the handler branches back to the driver, which in turn branches back to the program from which the original branch occurred.

Fortran Compiler

Fortran is most suitable for writing programs to solve mathematical problems. It bears no resemblance to the machine language of the computer, and so cannot be loaded into memory without first being translated. The Fortran compiler performs this translation. The result of this translation is called a relocatable object program. It contains the instruction sequence necessary for the computer to solve the original problem. It is called relocatable because it can be loaded into any area of memory by the relocatable loader program to be described later.

Assembler

The function of the Assembler program is to translate assembly language source programs into relocatable object programs. Assembly language is a convenient means of representing a machine language program. Mnemonics are used to represent each instruction, and symbols are used to identify addressed locations where necessary. There is almost a one-for-one correspondence between the source program and the instructions as they appear in core when the relocatable object program is loaded by the relocatable loader. The relocatable object program produced by the assembler is in the same format as that produced by the Fortran compiler; in fact, the loader cannot tell them apart.

Relocatable Loader

The relocatable loader has two functions. The first is to load into the area of memory specified by the operator a relocatable object program or load module, and the second is to establish linkages between a main program and its subprograms. A relocatable object program contains the instructions which make up the program and also a set of commands to the relocatable loader. The purpose of these commands is to describe the address field of each instruction. Those instructions which refer to other instructions or data locations within the same program have relative address fields. When the program is loaded into a section of memory other than that for which it was written, the starting address must be added to each instruction with a relative address field. Those instructions with absolute address fields are not modified (they refer to fixed locations in memory, are instructions with no address field, or are constants). A third possibility for the address field is that it be external. This is the case when an instruction in one program refers to the start of a subprogram. The loader must keep a

table of all such external references as the program is loaded. Later, when the subprograms are loaded and their starting addresses are known, the loader must go back and fill in the address field of each external reference instruction.

If the operator decides to have the loader store a load-module, then the relocatable loader must also store a table with the program. The function of the table is to list all instructions within the program with relative address fields. This makes it possible to tell after loading which instructions are location dependent. (This will be explained in more detail later.)

Absolute Dump

The function of the absolute dump is to produce an absolute object tape of a section of memory specified by the operator. Each location is dumped without modification, and there is no indication on the tape whether the instructions have relative address fields or not. Thus, such a program can only be loaded back into the same area of memory from which it came. The format of the absolute object program tape is different from that of the relocatable object tape, and it must be loaded by the absolute loader. Absolute object tapes represent a convenient way to keep a permanent record of such things as the system software programs. These tapes can be loaded by a fairly short and simple loader program.

Absolute Loader

The absolute loader loads into memory absolute object programs as produced by the absolute dump. The loader stores the program back into exactly the same locations from which it was originally dumped.

Load-Module Dump

The function of the load-module dump program is to dump onto paper tape a complete load module as loaded by the relocatable loader and modified by the programmer in the process of debugging. As the program is dumped, the relocation table which is part of the load module is examined to determine the address field type for each instruction. The starting address of the program is subtracted from those instructions with relative addresses, before dumping. The relocation table is included on tape with the program so that the load module loader can relocate the load module, on loading, to any part of memory. The main program and subprograms are dumped together as if they were one. The format of the load-module tape is different from those of the relocatable and absolute object tapes, and it must be loaded with the load-module loader.

Load-Module Loader

The function of the load-module loader is to load into memory load-module tapes dumped by the load-module dump program. The operator indicates to the loader the starting address where the program is to be loaded, and whether or not the relocation table is to be stored with the program. As the program is being loaded, the loader checks the relocation table (which is on the tape with the program) to determine the address field type of each instruction, and adds the starting address (as previously specified by the operator) to those which have relative addresses.

Debugging Aids

The following programs are included in the software system to facilitate the programmer's task of correcting a program. They require that the program to be corrected be in memory. They make extensive use of the powerful input/output facilities of the cathode-ray tube display and light-pen combination.

DAMM (Display And Modify Memory)

This program causes sixteen consecutive memory locations and their corresponding addresses to be displayed on the CRT. Using a keyboard to select the desired function and the light pen to indicate the word to be modified, the operator can alter any displayed memory location as he pleases. He can also select different sections of memory to be displayed. An additional feature allows the operator to transfer control to any other program in memory. Thus, after modifying a program with DAMM, the user can cause the program to be executed immediately to test his modification without halting the computer.

DAMM also allows the operator to make changes to the program section of a load module in memory without having to worry about the accompanying relocation table. It continuously updates the table as the operator makes changes to the program. The CRT display, keyboard, and light pen are used to make the changes.

DMWM (Display Memory With Mnemonics)

This program displays a section of memory in a fashion similar to DAMM; however, the mnemonic code for each displayed instruction is also displayed, making it easier for the operator to interpret the instructions. Using the keyboard, the operator can select different areas of memory for display. He can also cause the computer to branch to DAMM to modify, if desired, one of the locations just inspected by DMWM. DMWM, also at the option of the operator, can cause the displayed section of memory and its mnemonic equivalent to be the output to any other desired device. Thus a permanent record of a corrected program can be

obtained by using the paper tape punch, or the teletype machine, as the output device.

DBUG

This program allows software execution of a user program, either one instruction at a time, or sequentially. All registers and pertinent memory locations are displayed. Software equivalents of halt switches, and a memory protect system are also provided. It is not necessary, therefore, to halt the computer while executing a program one instruction at a time, allowing external interrupts to have free access to the machine. DBUG uses the CRT display as its primary input-output and control device.

Service Programs

Two other programs that are part of the software system facilitate making modifications to source programs on paper tape. One allows the operator to choose the format of both input and output from a number of possibilities. This facility can be used for copying paper tapes, copying from paper tape to magnetic tape, and so on. The other program is designed to make a paper tape look like a deck of cards; that is, to make it easy to insert and delete lines from a source language tape.

Software Protection Feature

A hardware feature of this computer (and of most others) is that each word has an additional bit over the 24, called the protect bit. If this bit is turned on (is 1, not 0), it indicates a protected location. At the operator's console is a switch which selects either the protected mode or the unprotected mode for the computer. If the computer is in the protected mode, then protected instructions can be executed normally, but unprotected instructions can only operate on other unprotected locations. If an unprotected instruction attempts to store in or branch to a protected location, an interrupt occurs. If the computer is in the unprotected mode, then there is no distinction between protected and unprotected locations. A protect bit can be set or reset under program control only if the computer is operating in the unprotected mode.

When programs which do not yet function properly are being tested, it frequently happens that they accidentally modify the contents of large areas of memory, destroying any programs there. To provide for rapid recovery from such accidental erasure of memory, the entire software system is stored on one roll of paper tape in the absolute format. The computer is normally operated in the protected mode with nearly all locations unprotected. Only a very small program, called the bootstrap loader, is protected. This short program can load the system software tape from the high-speed tape reader and thus restore the

system quickly. If the bootstrap loader is protected and every other location is not, and the computer is operating in the protected mode, this loader can never be accidentally destroyed. If it should be accidentally erased by the operator forgetting to place the computer in the protected mode, the operator would then have to enter the bootstrap loader manually from the console.

SOFTWARE SYSTEM APPLICATION

The application of the software system to development of a program can be illustrated by showing the steps involved in the transformation of an idea into a working program. These steps are illustrated in Fig. 2.

The programmer first converts his ideas into a flow-chart depicting the logical organization of the program. This is the most difficult part of the process; it requires that the programmer decide on a set of simple operations that will accomplish the purpose of the program.

Once the flow-chart has been drawn, the programmer must decide which is the language most suitable for its coding. If the program involves many mathematical operations, then Fortran would be the logical choice. If, on the other hand, the program involves data manipulation, control of peripheral devices, or similar operations, then it would be most easily coded in Assembly language. In either event, once coded, it must be converted to a form suitable for use by one of the computer's input devices. With the present system, this involves punching the source program on paper tape. The source program is then translated by either the Assembler or Compiler (depending on the coding of the source program). During this process, language errors are detected by the translating program and listed on the CRT display. The programmer then immediately corrects the source tape with the aid of one of the service programs, and the translation process is repeated. When all such errors have been removed, a relocatable object program is obtained.

The relocatable object program is loaded by the relocatable loader into a section of memory not occupied by any of the software system components. Any subprograms required by the main program are also loaded at this time, and the linkages between them are established automatically by the loader program. Generally, the programmer will choose to have the relocation table stored with the program (to result in a load module).

The next step is to try executing the program to determine if it performs as was originally intended by its author. If it does, then the programmer is finished provided that a relocatable object version of the program was his goal; if he prefers to have a load module on tape, then he can obtain one by using the

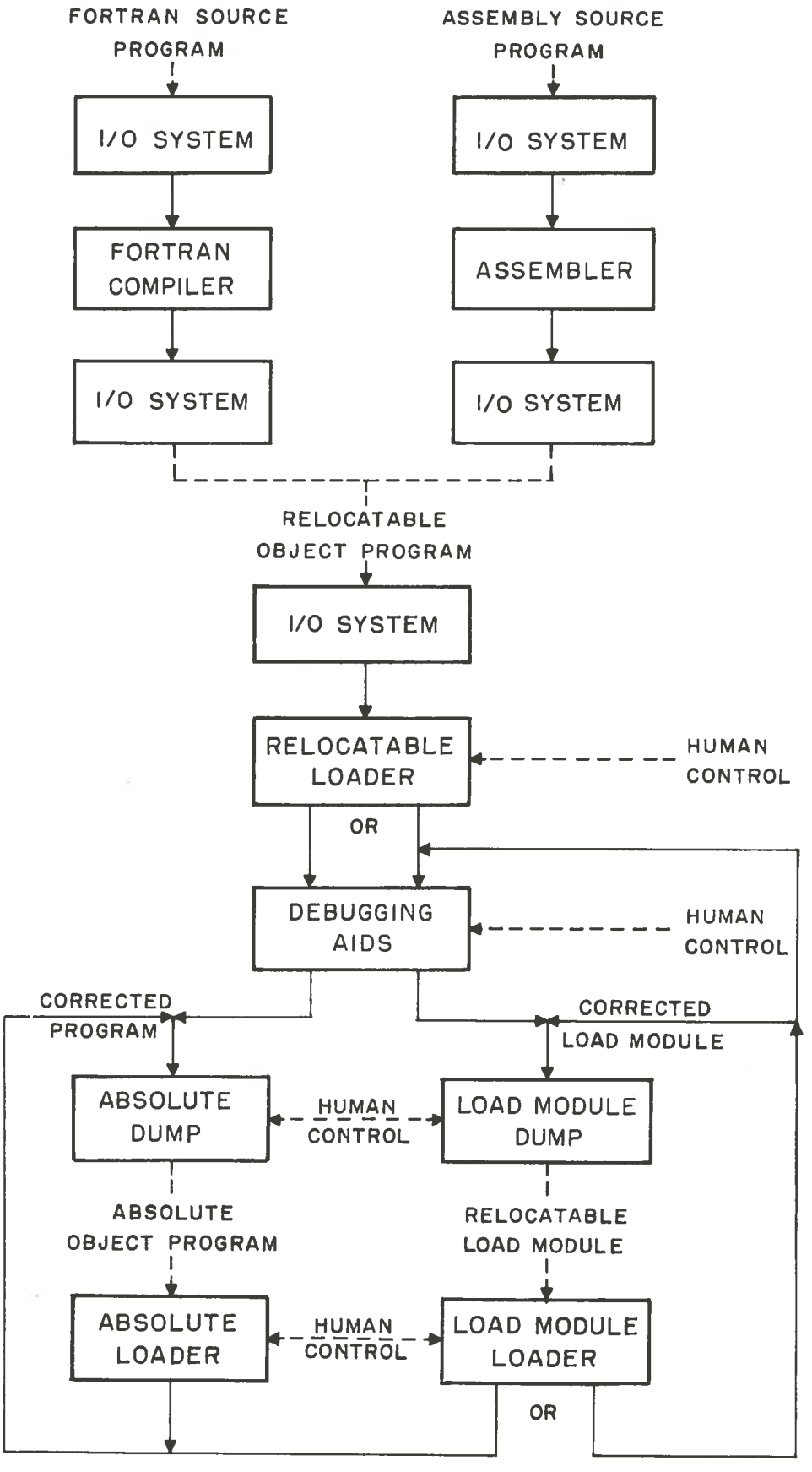

Fig. 2 Program development

load-module dump. If, as more usually happens, the program does not function as expected owing to some oversight of the programmer, then he must try to determine the reason. Quite often, the necessary modifications to the object program are of a minor nature and can easily be made to the load module in memory with the aid of the debugging programs. Once a correct load module exists in core, use of the load-module dump will produce a tape of the same thing. The load-module tape is now the finished version of the program; it can be loaded anywhere in memory with the load-module loader, when required. If the modifications required to correct the program are extensive, then it is usually easiest to modify the source tape directly, using the service aids, and begin again at the translation stage.

The software system, as described in this paper, has proved to be a useful tool to speed the development of programs.

Additional software is currently undergoing trial operation in the system. An executive program which runs continuously is used to initiate each program in the software package described above. Each of the system programs branches back to the executive, instead of halting, each time a job is finished. The function to be performed is indicated to the executive by the operator through the CRT display and light-pen combination.

The executive package also makes use of a permanently enabled hardware interrupt which returns control to the executive at any time by pressing a button. This allows recovery from various forms of failure such as an attempt to execute an instruction with an illegal op-code.

When the system operates under executive control, other users may cause their programs to be executed as a result of interrupt requests and so a form of multiprogramming with many simultaneous users is possible. This method has been used to allow on-line checkout of new peripheral equipment without disturbing normal program development.

BIBLIOGRAPHY

1. Ivan Flores. Computer Software. Prentice-Hall, 1965

ACKNOWLEDGMENT

The author has described the existing software system for the Data Systems Section research computer. He wishes to acknowledge that members of the Data Systems Section contributed programs and supervised the development of this system, and that this report includes many contributions from other members of the group.

APPENDIX

GLOSSARY OF COMPUTER TERMINOLOGY

Absolute object program - machine language program which can only be loaded into specific locations of memory, usually those from which it came.

Address - unique number assigned to represent a word or location in memory.

Assembler - program to translate a machine language source program in mnemonic form into a relocatable object program.

Bit - binary digit. Can be either 0 or 1.

Compiler - program to translate a Fortran source program into a relocatable object program.

CRT - cathode-ray tube. Similar to a television picture tube.

Debug - remove the errors from a program.

Executive program - program to control the operation of the computer and software system.

Flow chart - diagram consisting of blocks interconnected with arrows showing the simple steps of a program and the order in which they are performed.

Hardware - the computer and its associated peripheral devices.

Instruction - number stored in the memory which describes a simple operation to be performed by the computer.

Language - mode of expression used by humans in writing a program.

Light pen - pen-shaped device sensitive to light.

Loader - program which can read object programs from a peripheral device and store them in memory.

Load module - complete object program together with table listing those instructions with relative address fields (i.e., those which would have to be modified if the program were relocated elsewhere in memory).

Machine language - instructions which the computer can understand.

Object program - result of a source program being translated into machine language — may be in memory, punched on paper tape, etc.

Operating system - complicated set of programs to control automatically all operations of a large digital computer.

Peripheral device - piece of equipment connected to, but not an integral part of, the computer. Usually used for input or output functions. May also be an external storage unit.

Program - accumulation of machine language instructions or higher language statements which describe to the computer how to solve a particular problem or perform a certain task.

Register - temporary storage device capable of storing one word or less.

Relocatable object program - machine language program which can be loaded into any area of memory.

Software - accumulation of programs which control a computer.

Source program - program before translation into machine language.

Sub-program - program which performs a certain task, and is called by a main program each time performance of that task is required.

Teletype machine - A typewriter-like device which can be operated electronically.

Time-sharing - Simultaneous use (or apparently so) of one computer by more than one person or program.

Word - One addressable location of memory capable of storing a fixed number of bits. Can be used to store an instruction or data.