HONEYWELL EDP

SOFTWARE BULLETIN

SERIES 200

INTRODUCTION TO SERIES 200/OPERATING SYSTEM-MOD 1

SUBJECT: General Description of the Series 200/Operating System-Mod 1.

DATE: March 30, 1966

FILE NO. 122.0005.001C.0-$\overset{*}{\underline{258}}$

8787
10366
Printed in U. S. A.

*When ordering this publication please specify Title and Underscored portion of File Number.

FOREWORD

This bulletin introduces the Series 200/Operating System-Mod 1. Section I summarizes Honeywell's extensive experience in operating system design and describes the major benefits to be derived from the use of the Mod 1 system. A complete description of the over-all Mod 1 design is presented in Section II, followed by several examples of system operation. Appendix A contains a list of all Mod 1 publications to date, together with their respective DSI or order numbers.

TABLE OF CONTENTS

		Page
Section I	Introduction	1-1
	Honeywell's Operating System Experience	1-1
	Major Benefits of the Series 200/Operating System — Mod 1	1-2
	Turnaround Time is Minimized	1-2
	Throughput is Maximized	1-2
	Flexibility and Orderly Growth Potential are Provided	1-3
	Maximum Use is Made of Memory and Peripheral Units	1-3
	Overhead is Minimized	1-3
	Mod 1 Assists all Members of the Data Processing Team	1-3
	The Operator	1-3
	The Programmer	1-3
	The Installation Manager	1-3
	The Series 200/Operating System — Mod 1	1-4
Section II	Series 200/Operating System — Mod 1	2-1
	Processing Structure	2-1
	System Functions	2-2
	Program Preparation and Maintenance	2-2
	Language Processing Functions	2-2
	Program Editing and Maintenance Functions	2-6
	Program Control	2-8
	Operation Control Functions	2-8
	Input/Output Control Functions	2-9
	Program Test Functions	2-11
	Utility	2-13
	Data Transcription and Editing Functions	2-13
	Mathematical Processing Functions	2-14
	System Operation	2-15
	Operation With Source Programs on More Than One BRT	2-16
	Combining Source Programs for Operation from One BRT	2-16
Appendix A	Mod 1 Publications	A-1
	Program Preparation and Maintenance	A-1
	Program Control	A-1
	Utility	A-1

LIST OF ILLUSTRATIONS

Figure 2-1.	Mod 1 Functions	2-3
Figure 2-2.	Operation With Source Programs on More Than One BRT	2-18
Figure 2-3.	Combining Source Programs for Operation from One BRT	2-19
Figure A-1.	Mod 1 Publications	A-2

LIST OF TABLES

Table 2-1.	Program Preparation and Maintenance	2-7
Table 2-2.	Program Control Programs	2-10
Table 2-3.	Utility Programs	2-14

SECTION I
INTRODUCTION

The Series 200/Operating System-Mod 1 may be viewed as a framework within which all of the user's data processing jobs can be scheduled and performed. More specifically, this advanced operating system is a comprehensive set of language processing and service programs, executed under the supervision and coordination of an integrated group of control routines. Designed for implementation in a wide range of environments — in differing stages of growth — the Series 200/Operating System-Mod 1 brings to the user a powerful tool designed to minimize turnaround time and maximize throughput for his computer installation.

HONEYWELL'S OPERATING SYSTEM EXPERIENCE

While other computer manufacturers have been announcing the introduction of operating systems to their product lines, Honeywell has been delivering its third-generation operating system to customers.

The first Honeywell operating system was created for the D-1000 computer in 1957. The D-1000 system contained a monitor program and extensive program debugging facilities and was the first system of it's kind to employ file updating techniques for program checkout operations. An advanced Program Test System was developed for the H-800 medium scale computer in 1959. The system included automatic checkout and debugging facilities for production operations — manual operations were thus limited to standard loading and starting commands for an entire sequence of programs.

The Executive System released by Honeywell, in 1960, was the first multiprogram operating system. It was developed to automate further the execution of several sets of programs in parallel. Since multiprogram execution was performed automatically by the hardware, the role of the operator was limited to communication with the operating system and peripheral setup. The Executive System consisted of scheduling and execution phases. Scheduling included the performance of all of the tasks which must be accomplished prior to execution, such as program selection, hardware allocation, program relocation, and preparation of a detailed schedule summary. Execution included supervised performance of the prepared program schedule by a small monitor which sequenced programs and informed the operator of the progress of the run, issuing detailed instructions when manual intervention was required.

The operating system concept was expanded significantly by Honeywell's introduction of the more advanced Admiral multiprogram operating system in 1963. The Admiral Operating System was designed for H-800/1800 computers and incorporated a dynamic scheduling facility which enabled the user to stack job requests in a queue and obtain automatic processing with optimal utilization of all system components. The Admiral scheduling function achieved a constant saturation of processing facilities while satisfying the external priority constraints established by the user. Because of it's superior dynamic multiprogram scheduling capability, Admiral is still one of the most advanced operating systems in field use.

The experience which has accumulated from the development of three generations of operating systems has changed the Honeywell concept of operating systems from a group of supervisory and utility routines to a computer management system that now encompasses all control, language processing, and service functions. Many of the functions which were formerly regarded as the separate domains of programmer, operator, and installation manager have been combined, and the responsibility for efficient data processing has been given to the operating system. This evolution has been greatly enhanced by the concurrent development of new hardware features which can be most effectively managed on a total system basis rather than on an individual component basis. The hardware and software functions have been designed to complement one another and operate as a single entity which will maximize throughput, minimize turnaround time, and generally assist in the activities performed by the various members of the data processing team.

MAJOR BENEFITS OF THE SERIES 200/OPERATING SYSTEM-MOD1

Turnaround Time is Minimized

One of the best measurements of the value of the Series 200/Operating System-Mod 1 is its turnaround time (i. e., the interval which exists between the time a job is submitted for processing and the time at which the results are returned). Here are two of the ways in which its extremely low turnaround time is achieved:

1. All required operations receive maximum automation; the extent of human participation is limited and controlled.
2. Delays are eliminated through the automatic processing of jobs from beginning to end on a single system.

Throughput is Maximized

Another measure of the value of Mod 1 is throughput (i. e., the total amount of work which the system can perform in a given period of time). Here are some of the ways in which the operating system maximizes throughput:

1. All available system resources can be efficiently allocated.
2. Idle system time and job setup time can be reduced to an absolute minimum.

3. Facilities are provided for uninterrupted processing of a continuous stream of jobs.

Flexibility and Orderly Growth Potential are Provided

1. Use of an open-end design facilitates inclusion of the user's programs and data in the system.
2. The modular design of the operating system allows a wide variety of jobs to be devised using the system's facilities.
3. Program compatibility is assured as the system grows.

Maximum Use is Made of Memory and Peripheral Units

Through multiprogramming (the capability to execute two programs simultaneously), the Series 200/Operating System-Mod 1 is able to use central processor memory and peripheral units to maximum advantage.

Overhead is Minimized

The characteristic fixed overhead — which makes many operating systems economically unfeasible — is eliminated by the modular design of the Series 200/Operating System-Mod 1. This operating system can be implemented in a wide variety of environments and in differing stages of growth. It enables the user to choose only those functions and the degree of centralized control for these functions which are specifically oriented to his equipment configuration thus sparing him the cost of unnecessary functions and features.

Mod 1 Assists All Members of the Data Processing Team

THE OPERATOR

1. He is relieved of detailed and burdensome execution supervision.
2, He uses standardized operating procedures.
3. He is provided with instructions which are precise and to the point.
4. He has facilities which enable him to communicate with and direct the system.

THE PROGRAMMER

1. His own programs as well as library programs and systems programs may be run as integral parts of the operating system.
2. He is provided with facilities to enable programs and data to be easily stored, easily tested, easily modified, and easily referenced.
3. He is supplied with a variety of programming aids which enable him to express the current problem in an easily understood language.

THE INSTALLATION MANAGER

1. His system programs can be maintained simply and economically.
2. He can apportion the programming effort.

THE SERIES 200/OPERATING SYSTEM-MOD 1

Honeywell's experience, gained in the development of operating systems for earlier generations of computers, has been combined with the latest innovations in EDP in order to provide a new and powerful operating system for the Series 200. The resulting product is a system which affords users a full complement of automatic operating functions within truly minimal equipment requirements. The user may select among several types of program preparation and maintenance functions, data transcription and editing functions and utility functions. To these functions, comprehensive control functions are added in order to provide the user with increased system throughput. The user may make the trade-off decisions which enable him to attain fast turnaround time or more efficient use of machine facilities, i. e., he may choose to execute programs either in the multiprogramming mode or in the serial mode. He may also bridge the gap between program setup and execution by means of space sharing and/or time sharing and thereby eliminate idle system time.

The user is free to increase the quality of service in his installation because he determines the most efficient type of work load scheduling and operating mode. Non-productive computer time is reduced to a minimum because the extent of required operator intervention is reduced. Emergency conditions are indicated by means of standardized messages and handled via standard emergency procedures, thus reducing the necessity for a highly trained, specialized operations staff. Maintenance of the system is both simple and economical. Initial system generation is a flexible, selective process by which the user incorporates into the system only the functions which are required for his level of processing; maintenance of the system and user programs thereafter is facilitated by means of simple updating runs.

Both hardware and software are modular in structure, compatible in function, and flexible in implementation. Both hardware and software consist of integrated modules: hardware modlues consist of sets of circuitry required for a particular system function, thus simplifying the expansion of a configuration; software consists of independent modules that accomplish specific functions under the control of the operating system, thus facilitating a building block approach to program development. Both hardware and software contain compatible elements: any model of the Series 200 can utilize all of the wide range of peripheral devices and control units with any of the central processors in the Series 200 line, and source programs from any one of the Series 200 computers can generally be processed on all other (larger) models if there is enough memory capacity and peripheral equipment. Upward compatibility of programs exists via a single machine language for all models of the Series 200. Hardware and software are both flexible in implementation: because of the interchangability of devices, control units, and central processors and because of the selective capability of system functions. The user has complete freedom in the choice of both hardware and software in order to obtain the exact combination of facilities which is best suited to his data processing requirements.

The description of the operating system is presented in terms of several groups of functions which, when taken together, contribute to the total processing capability of the system. The groups are:

1. Program Preparation and Maintenance Functions: The user is provided with a comprehensive language processing capability by means of the assemblers, compilers, and conversion programs. In addition, facilities are included for the creation and maintenance of symbolic and machine-language program files.

2. Program Execution Functions: Includes operation control, input/output control, and program test functions which are related to the execution of both systems and object programs.

3. Utility Functions: All functions which are used in support of the above primary functions are discussed as utility functions or separately described according to the special purpose(s) which they perform as part of the operating system.

SECTION II
SERIES 200/OPERATING SYSTEM — MOD 1

The Series 200/Operating System — Mod 1 provides the user with a wide range of processing functions designed to enhance the throughput capabilities of medium-scale Series 200 computers. Primarily tape-oriented, the Mod 1 functions incorporate a design flexibility that permits both independent and semi-centralized operations to be performed; overhead is thereby reduced to a minimum. This flexibility is exemplified in the ability to intermix and process (at execution time) programs written in different source languages. The maintenance functions of Mod 1 are equally applicable to both the systems programs furnished by Honeywell and to the production programs generated by the user; in addition, the programs may be maintained at both a source-language level and at the machine-language level.

Mod 1 is designed for Series 200 computers containing a minimum of 12K of core memory and from three to six magnetic tape units. The flexibility of the system enables efficient use to be made of a variety of peripheral devices — including mass storage, communication, paper tape, and punched card.

PROCESSING STRUCTURE

Fundamental to the design of Mod 1 is functional program modularity. This modularity is achieved by the segmentation of programs into loading units which accomplish specific functions under operating system control. (By definition, a "loading unit" is that portion of code which is found and loaded as the result of a single call to the Loader-Monitor.) Thus, although the basic logical or processing unit is the program, a program is normally segmented into several loading units to provide operational convenience and flexibility.

Related programs can be combined; this capability provides the freedom of a building-block approach to the development of production jobs. The sequencing of related programs may be accomplished automatically (through the common interface of the operating system) by either (1) internal directors (instructions incorporated in the user's programs) or (2) external directors (console call cards). The latter method does not require cross referencing between programs; hence, the user is free of operating considerations at the coding level.

Systems programs exist as independent units which can be called in any sequence as they are needed to fulfill the user's requirements. Since the user's programs as well as systems programs can be combined into a single executable file, both categories of programs can be called as they are needed.

An extensive effort has been expended by Honeywell in the design of Mod 1 — thereby producing an operating system having maximum integration of software with semi-centralized operations. The user's need for concern about systems functions has thus been eliminated, allowing him to concentrate on the problem-solving tasks pertinent to his specific application. This objective of an optimal programming/operations relationship has been attained through programming flexibility, simplified program testing, and standardized systems procedures. The Series 200 installation may thus be used with maximum efficiency for a variety of data processing requirements.

SYSTEM FUNCTIONS

As mentioned in Section I, Mod 1 is composed of several groups of functions: Program Preparation and Maintenance Functions, Program Control Functions, and Utility Functions. The relationship between these groups — which together make up the total processing capability of the system — is illustrated in Figure 2-1.

Program Preparation and Maintenance

In Mod 1, the functions of Program Preparation and Maintenance are distributed among (1) language processing functions and (2) program editing and maintenance functions. These functions are described below.

LANGUAGE PROCESSING FUNCTIONS

These functions consist of programs which provide the user with means to translate source programs into a single machine-language format. (The output from the various language processors may therefore be combined to fulfill the user's processing requirements.) Four types of language processors are provided — each with specific versions tailored to the user's specific requirements. These include:

1. Assemblers: Translate symbolic source (assembly) language into machine language;
2. COBOL Compilers: Translate commercial (COBOL) language into machine language;
3. Fortran Compilers: Translate scientific (Fortran) language into machine language; and
4. Conversion Translators: Translate competitive languages into Honeywell language on both a source-language and a machine-language level.

The user can select the method (or methods) of language processing best suited to his

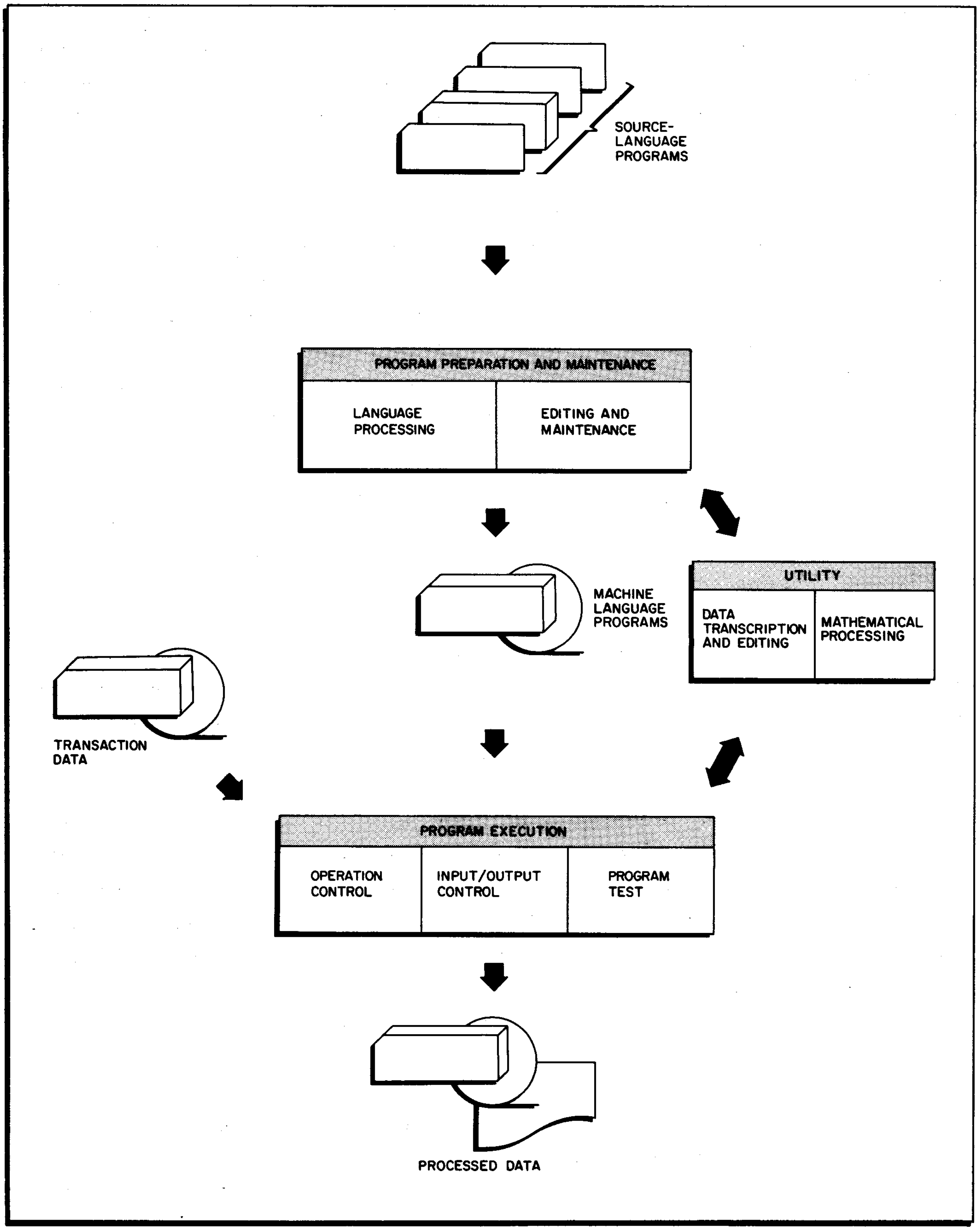


Figure 2-1. Mod 1 Functions

requirements in accordance with the nature of his job, experience of his personnel, compatibility requirements with other equipment, or personal preference. Each of the language processing subsystems produces executable programs in standard format; thus, programs produced by any of the subsystems can be combined for execution in a single sequential operation.

Assemblers

Mod 1 assemblers translate programs written in Easycoder symbolic language into machine-language object programs — taking full advantage of magnetic tape configurations to provide flexible handling and fast processing of symbolic source programs. The results are stored on magnetic tape in either symbolic or binary executable form. An efficient, flexible, and easy-to-use symbolic language, Easycoder permits the user to code source programs in convenient, understandable terms using easily remembered mnemonic operation codes. With Easycoder, memory location addresses may be denoted by absolute decimal numbers or by symbolic tags, and the value of an operand may be expressed as a literal.

The assemblers provide the programmer with other very useful tools. One of these tools — symbolic program analysis — enables any Series 200 program that is coded in Easycoder symbolic language to be analyzed. In the process of analysis, symbolic tags, references (to each tag, to index registers, and to absolute addresses), and calls to library routines are extracted from the symbolic input program to be analyzed. This information is processed to produce a printed listing which is arranged in alphanumeric order so that all information about a particular tag, absolute location, or library routine appears grouped in one place on the listing.

Another very useful tool — symbolic library processing — is provided as an extension of the assembly language. Through the use of "macro" calls written into the user's program, the library processing element enables the programmer to efficiently incorporate precoded assembly-language routines into his program. In addition, the called routine is specialized to the particular program employing it in accordance with parameters specified by the programmer. The library routines may be Honeywell-supplied (e.g., Input/Output routines) and/or they may be written by the user to meet the special requirements of his installation.

COBOL Compilers

Programs written in COBOL language (a standardized, business-oriented subset of English) are translated into machine language by the Mod 1 COBOL compilers. Series 200 COBOL language is a convenient, relatively machine-independent method of expressing a data processing problem to a computer.

Series 200 COBOL compilers possess several important operating features. For example, these compilers can accept batched source programs and can operate in a batch-compile, load-and-go mode. Source-language programs can be maintained in magnetic tape libraries to facilitate program correction during checkout. A variety of testing and debugging aids is provided with all COBOL compilers, such as dynamic and static dumping capability, English-language diagnostics, memory mapping, and test data distribution. In the interest of operational flexibility, the addresses of peripheral devices can be assigned at object time to allow the use of a single program with a variety of different peripheral arrays.

Fortran Compilers

The Mod 1 Fortran compiler features an implementation of Fortran as proposed by the American Standards committee on March 10, 1965 with several additional language and debugging features. This compiler translates, into machine language, programs which have been expressed in a format similar to algebraic equations. (Programs are written directly as algebraic expressions and arithmetic statements.) Additional statements, such as transfer, decision, indexing, and input/output statements, control the processing of the algebraic expressions.

Translated programs can be combined with other previously compiled and assembled programs and immediately executed, thereby obtaining rapid results.

Conversion Translators

The Honeywell Liberator concept enables competitive languages to be translated into Honeywell language on both the symbolic-language and machine-language levels. Users of a number of older systems are thus permitted to enjoy the benefits of Series 200 processors — without the cost and effort of reprogramming. This concept has many facets including compatibility of programming languages, data formats, and peripheral input/output devices.

An example of the effectiveness of the Liberator concept may be seen in Easytran. An input consisting of symbolic source programs, written in SPS and/or Autocoder language, is completely analyzed and then translated statement by statement. During this process, most symbolic statements are replaced on a one-for-one basis with equivalent Easycoder statements. Those statements which have no direct Easycoder equivalent are replaced either with in-line macro-coding or with calls to Easytran subroutines which perform the desired functions; those whose functions are not required by Honeywell hardware are deleted.

A list of the language processing programs, together with a brief description of each program, may be found in Table 2-1, page 2-7.

PROGRAM EDITING AND MAINTENANCE FUNCTIONS

Processes such as storing, modifying, and maintaining assembly-language and binary-executable programs comes under the heading of program editing and maintenance functions. These functions enable programs to be selected and ordered — in the sequence best suited to the specific requirements of the job — to create processing run tapes which contain only the required systems and processing programs. Also worthy of note is that for maintenance operations, assembly-language input is handled using a method that requires only new or changed source-language statements as input.

The processes of editing and maintaining symbolic programs include program updating and program selection. The updating process provides for the maintenance of a symbolic program tape file through the correction of individual programs, the addition of new programs, and the deletion of unwanted programs. Likewise, the selection process provides for the selection of specified programs from a symbolic program tape and the recording of these programs on a binary run tape. Also included among the processes for manipulating programs in their symbolic form is the symbolic program tape merge process which provides easier and faster handling of programs that are stacked on symbolic program tapes (SPT). This process performs the selection and extraction of programmer-designated programs from the input SPT's (as many as four) and writes them on a new SPT. Thus, it is possible for the user to consolidate programs stored on several SPT's onto one master tape with the programs arranged in any desired order. However, the most important aspect of this process is that the programs may be selected, copied, and rearranged onto the master tape without having to reassemble any of the programs in the process.

Similar to the processes for editing and maintaining symbolic programs, the editing and maintenance processes for binary-executable programs also include program updating and program selection. These processes accept programs in absolute binary form as produced by the Easycoder assembly, the COBOL compiler, or the Fortran compiler subsystems and enter the programs into an updatable library. The programs in the library may be updated using octal corrections or they may be replaced or deleted; new programs may be added to the library; and a selected run tape can be produced containing specified programs selected from the library and ordered in any fashion.

Also included among the program editing and maintenance functions are the process for creating a program file on a drum and the process for producing punched-card programs from a binary run tape. The program file on a drum consists of a drum loader (optional), a drum monitor (optional), the user's object programs to be loaded into memory by the drum loader or monitor, and a program directory of the file. The process for creating the drum file accepts object programs produced by any of the language processing subsystems and converts this input into a format acceptable to the drum loader and monitor. This process may thus be used to produce a self-loading drum file or merely for program storage.

Again, system flexibility is expanded by the editing process for converting programs written on a binary run tape (BRT) into a BRT punched-card format. By this process, the input programs which are loaded and monitored under the control of the tape loader-monitor are reformated for loading and monitoring by the card loader-monitor.

A list of the program editing and maintenance programs, together with a brief description of each program, may be found in Table 2-1, below.

Table 2-1. Program Preparation and Maintenance

	Program Name	Description
LANGUAGE PROCESSING	Easycoder Assembler C	Translates Easycoder symbolic language into Series 200 machine language.
	Easycoder Assembler D	Translates Easycoder symbolic language into Series 200 machine language.
	Analyzer C	Prepares cross reference list of symbolic references in an Easycoder program.
	Library Processor C	Specializes macro routines.
	COBOL Compiler D	Translates COBOL source language into machine language.
	COBOL Compiler H	Contains all the features of COBOL D with the added ability to address up to 256K at object time.
	Fortran Compiler D	Translates Fortran source language into machine language.
	Fortran Compiler H	Translates Fortran source language into machine language.
	Easytran Symbolic Translator C	Translates Autocoder or SPS into Easycoder C symbolic language.
	Easytran Program Modifier C	Produces an Easycoder C symbolic program from an Easycoder A symbolic program which previously resulted from translation using either Easytran 1401 or Easytran Symbolic Translator B.
	Easytran Symbolic Translator D	Translates Autocoder or SPS into Easycoder C symbolic language.

Table 2-1 (Cont). Program Preparation and Maintenance

	Program Name	Description
PROGRAM EDITING AND MAINTENANCE	Update and Select C	Updates master file of programs in BRT format.
	Update and Select D	Updates master file of programs in BRT format.
	SPT Merge C	Merges and/or reorders symbolic program tapes.
	Drum Program Store C	Stores Mod 1 programs on the drum; also enables direct loading to be performed from the drum.
	BRT Punch C	Punches cards in BRT format from a binary run tape.

Program Control

The Program Control functions include (1) operation control, (2) input/output control, and (3) program test. The central function — operation control — includes loading, monitoring, program sequencing, and other related elements.

OPERATION CONTROL FUNCTIONS

The operation control functions complement the Series 200 hardware and provide a flexibility that permits the user to operate in a variety of environments including those consisting of all tapes or tapes mixed with mass storage. This hardware/software complement also provides the necessary interrupt capabilities which enable the user to take full advantage of multiprogramming and communication equipment. It logically follows that peripheral data conversion or real-time operations may be performed concurrent with a major data processing job.

Loading is accomplished with highly efficient usage of core memory. This is possible since (1) only the loader resides in memory, and (2) the object programs may be segmented into optimum-size loading units (only the required portion of the program need thus be in memory at any one time). The loading process may operate from anyone of several media such as punched cards, magnetic tape, and mass storage. Regardless of the medium employed, the loading process retains the same functional characteristics and incorporates features permitting the most practical and convenient operation according to the size and type of computer installation.

Another important aspect of the operation control function is the ability to process a sequence of programs — with little or no operator intervention. This ability enables a series of logically connected programs or "jobs" to be processed. The programs which comprise the job are often independently generated and are made compatible for execution through the common interface of the operating system. Thus, a typical job may be a mixture of programs derived from the various language processing subsystems, systems routines, and common library routines. User-written programs, library routines, and systems programs can be combined, ordered, and run as an integral part of the operating system — allowing full advantage to be taken of Mod 1's automatic program loading and sequencing provisions.

The interrupt control process increases throughput by effective management of the simultaneous operation of programs that share available central processor cycles. More specifically, all machine cycles are alloted to a foreground program when operating in the interrupt mode until the foreground program issues a peripheral order. While a peripheral order of the foreground program is in progress, all machine cycles are allowed to a background program. An interrupt generated by the completion of the peripheral order causes allocation of machine cycles to be returned to the foreground program. The interrupt control process enables background programs to be sequenced independently of foreground programs, and in the same manner, foreground programs may be sequenced independently of background programs.

In addition to the processes mentioned above, the operation control function provides the facility for "floating" the loader-monitor to the end of the upper 4K memory bank. Also, a process is included for communicating with the operator via printed or typed instructions and for formatting and writing headings for reports.

A list of the operation control programs, together with a brief description of each program, may be found in Table 2-2, page 2-10.

INPUT/OUTPUT CONTROL FUNCTIONS

The Input/Output control functions consist of a set of macro routines which may be specialized and incorporated into a user's data processing system. These routines facilitate such operations as checking labels, checking file identification, and checking for read/write errors. The I/O control functions include processes which manage the standard input/output operations for magnetic tape, punched card, printer, drum, and console in such a way that the need for writing detailed and exhaustive input and output coding is eliminated.

The tape and terminal I/O process reads and writes files, blocks and unblocks records, labels tape files, checks for errors, and handles all the programming procedures necessary to complete peripheral operations. This process incorporates channel-test control procedures which permit the user to take advantage of the Series 200 simultaneity; namely, it automatically tests for the availability of a read/write channel and, upon finding an available channel, uses it for data transfer. A distinct advantage of the tape and terminal I/O process is that it allows the programmer to concentrate on the handling of items (logical groups of sequential fields within a record) rather than on handling whole records. It handles either blocked or unblocked records which, in turn, may contain either fixed-length or variable-length items. (A blocked

Table 2-2. Program Control Programs

	Program Name	Description
OPERATION CONTROL	Tape Loader-Monitor C	Locates programs on a Binary Run Tape and loads them into memory.
	Floating Tape Loader-Monitor C	Locates and loads programs from a Binary Run Tape.
	Drum Bootstrap Loader C	Used primarily to load Drum Monitor C into memory; however, it can load any program from the drum file.
	Drum Monitor C	Searches for, loads, and starts Mod 1 programs previously stored on a drum.
	Drum Interrogation, Alteration, and Loader C	Unloads, restores, edits, corrects, locates, clears, or compares drum data.
	Interrupt Control D	Controls execution of one background and one foreground (interrupt) program.
	Restarts C	Records contents of memory and position of tapes and later restores them.
INPUT/OUTPUT CONTROL	1/2-Inch Tape I/O C	Process all Honeywell or IBM 1400 Series tape files in any parity.
	1/2-Inch Tape and Terminal I/O C	Processes all Autocoder/IOCS files.
	Drum I/O C	Transfers fixed-or variable-length data between main memory and drums.
	Console I/O C	Processes messages (any format) which are input to (or output from) the console typewriter (operating as a peripheral unit).
	Communications I/O C	Provides communication capability via I/O macro routines.
PROGRAM TEST	Program Test Control C	Controls program checkout.
	Octal Correction C	Makes octal corrections to a program in memory.
	Memory Dump C	Edits contents of memory according to user's specifications.
	Tape Dump C	Edits contens of tape(s) according to user's specifications.
	Test Data Generator C	Prepares a data tape (or tapes) according to user's specifications.

record contains two or more items grouped together to form the record; an unblocked record contains only one item per record.) Thus, although all the records in a given file must be of the same form, the tape and terminal I/O process can process several different type files in the same program, including magnetic tape, punched card, or printer operations.

The drum I/O process, similar to the tape and terminal I/O process, makes it possible for the programmer to handle drum input/output operations with simple action macro statements. Each action macro statement produces the appropriate machine instructions which cause the desired input/output operation to be performed. For example, to access the next item in a drum file being read, the programmer merely writes a #GET statement in the source program; to obtain the address of the next available output location for a processed item, he writes a #PUT statement. Among the attributes of this process are the reduction of programming errors through the use of tried and proven methods, the standardization of data handling to provide the most efficient use of the Series 200 system, and the freeing of the programmer's time and attention for concentration on processing data, rather than on methods of accessing the data on the drum.

The console I/O process handles alphanumeric, decimal, and octal typeouts and typeins on the console typewriter. This process makes it possible for the programmer to communicate special instructions to the operator at the appropriate time during a program run by means of two types of macro statements which are easily incorporated into his program. The console I/O process also incorporates the use of standard messages and halts.

The input/output control processes are specialized as required to the applicable individual programs at the source-langauge level. This eliminates the necessity of having very general and diversified processes to handle the variety of peripheral devices available for the many equipment configurations, and also eliminates the requirement for tailoring of programs at execution time along with subsequent loading and linking of the necessary routines. Thus, these inefficiencies are avoided by specializing program requirements at the source-language level so that only those processes required by the object program are included.

A list of input/output control programs, together with a brief description of each program, may be found in Table 2-2, page 2-10.

PROGRAM TEST FUNCTIONS

The program test functions include several processes which may be used either separately or as part of an automatic checkout subsystem. These processes provide the following:

1. Automatic sequencing from one program to the next;

2. Printouts of messages and operator instructions.
3. Generation of test data;
4. Octal correction (patching) of programs;
5. Dynamic, terminal, and emergency memory dumps; and
6. Tape dumps.

The automatic checkout subsystem performs sequential checkout of several programs as directed by the test director deck. The ability is provided to process several stacked programs, with little or no operator intervention, and automatically produce the documentation necessary to evaluate the programs being tested. The subsystem processes may be performed before, during, or after object program execution. This action provides automatic testing by (1) reading an input card file of ordered test director decks (that call in each test process and object program from tape), and (2) executing the test processes and object programs in the specified sequence.

NOTE: Each of the test processes listed above may be performed separately. Some of these processes are outlined in more detail below.

Through the octal correction or patching process, the programmer is able to make specified octal corrections (or changes in the object program) in main memory at program execution time. The changes occur only in memory and do not affect the object program as stored on the binary run tape (BRT). This process causes the object program to be loaded and the applicable octal correction cards to be read. The corrections or changes indicated on these cards are then made to the specified locations of the object program, and the object program is automatically started.

The memory dump process edits and prints the contents of core memory (both data and punctuation bits) within limits specified at the time of execution. This process incorporates dynamic dumping capabilities as well as conventional static (terminal) dumping provisions. Dynamic dumping is defined here as the dumping of information while a program operates at high speed. Since the user can get a picture of how his program performed during execution by printing the contents of selected high-speed memory areas, the problem of locating errors is simplified. This process can also be used to provide emergency memory dumps if difficulty is encountered during the testing of object programs.

The tape dump process positions magnetic tape files and edits and prints the contents of these files. During a single operation, the tape dump process can handle up to 99,999 records having any record length up to the maximum size of 1,100 characters per record. The flexibility of this process is greatly enhanced by its ability to use either dynamic or independent operation, each of which may be specialized to fulfill the precise requirements of the user. For independent operation, this process is executed under operator control independently of other operations; whereas, for dynamic operation, the process is executed under control of the user's object program (or in conjunction with the automatic checkout subsystem).

A list of the program test programs, together with a brief description of each program, may be found in Table 2-2, page 2-10.

Utility

Mod 1 contains an extensive set of utility functions which provide a variety of transcription, editing, and mathematical processes.

DATA TRANSCRIPTION AND EDITING FUNCTIONS

Included among the data transcription and editing functions are such diverse processes as tape handling, media conversion, report generating, tabulator simulating, and sorting and collating of data stored on magnetic tape.

The Polyphase sorting process, first successfully implemented for computer processing by Honeywell, permits sorting with as few as three tape drives while minimizing the number of passes required over the data. Other processes offer facilities for collating up to five sorted magnetic tape files. The sorting process also provides the added advantages of read-backward Polyphase sorting and the ability to handle variable-length records. In addition, the sort process can be automatically incorporated into a series of related operations by coding the preceding program to establish the desired sort parameter values before transferring control to the sort process. Honeywell's advanced sorting techniques are also applied to the sorting of data in mass storage media.

Automatic creation of reports — according to user specifications — is provided by the report generated process. From an input consisting of programmer-specified parameters which define control fields and report lines, the report generator process produces a symbolic program. The assembled version of this program accepts raw data from cards or tape, edits it, and generates the desired reports.

The simultaneous media conversion process controls the transfer of data between pairs of devices such as magnetic tape drives, punched card equipment, paper tape equipment, and printers. This process increases the efficiency of media conversion processing by taking full advantage of simultaneity of Series 200. The user may also incorporate own-coding routines into the media conversion process for performing functions such as editing and unblocking of records.

A list of the data transcription and editing programs, together with a brief description of each program, may be found in Table 2-3, below.

MATHEMATICAL PROCESSING FUNCTIONS

The mathematical processing functions provide the Series 200 scientific users with an extensive library of scientifically oriented processes which complement the capabilities of the Fortran compilers. This library includes the usual basic Fortran routines, such as square root, exponential, trigonometric, and logarithmic functions, as well as matrix, statistical, and other more comprehensive processes. All the processes in this library can be used with or without the scientific hardware option.

A list of the mathematical processing programs, together with a brief description of each program, may be found in Table 2-3, below.

Table 2-3. Utility Programs

	Program Name	Description
DATA TRANSCRIPTION AND EDITING	Tape Handling Routine C	Performs positioning, copying, and editing operations on 1/2- or 3/4-inch magnetic tape.
	Simultaneous Media Conversion C	Performs from one to three conversions (between magnetic tape and cards, printer, or paper tape) simultaneously.
	Data Conversion C	A set of conversion routines for magnetic tape, cards, and printer.
	Tape Sort C	Reads backward as well as forward, taking advantage of Polyphase sorting techniques developed by Honeywell; can sort on six tape units in ascending or descending sequence.
	Tape Collate C	Collates five sorted tape files in ascending or descending sequence.
	Tape Sort C (V)	Same as Tape Sort C; handles variable-length items.
	Tape Collate C (V)	Same as Tape Collate C; handles variable-length items.
	Drum Sort C	Sorts fixed- or variable-length items in ascending or descending sequence.
MATHEMATICAL PROCESSING	Basic Math Functions C	Consists of a floating point package.
	Basic Math Fortran Functions D	Provides math functions for Fortran D systems.
	Modular Stitistics Package	Written in Fortran; provides such functions as linear regression and stepwise regression.
	Statistics Package D	Provides such functions as multiple regression and exponential regression.
	Matrix Package D	Provides such functions as matrix inversion and solution of simultaneous linear equations.
	Linear Programming D	Employs the revised simplex method for finding solutions to small problems which are contained in the available memory.

SYSTEM OPERATION

Mod 1 operations are controlled by operating directors. These directors may be specified either (1) internally using programmed calls or (2) externally through the use of console call cards or by manual entry through the console or control panel. Thus, in addition to the searching for, loading, and starting of the necessary routines to fulfill normal processing requirements, system control is provided for maintaining communications between the various components.

External directors are read from a peripheral device such as a card reader, or they may be entered from the console (or control panel). The directors specify to the operating system those routines necessary to fulfill the designated function; thus, only the required routines are read into memory and executed. When execution is completed, control is returned to the operator who may then specify that another director, to perform any one of a variety of functions in any order desired, is to be read. Automatic sequencing of programs may also be accomplished by directing the operating system to continue to read directors and execute the specified routines until the job is completed.

Internal directors are functionally identical to external directors. However, they differ in that they activate the operating system through programmed instructions which insert the proper information into the systems communications areas and then transfer control to the system. In this way, a predetermined series of routines may be performed without operator intervention or system halts. At the completion of the series, control may again be returned to the operator who is then free to initiate additional processing using external directors.

Through the use of external and internal directors, the user's object programs become an integral part of the operating system. Systems routines as well as the user's routines are thereby readily available and can be called and executed as required — assuring total, efficient processing of data to meet the user's requirements.

Several methods of operation may be employed using Mod 1. For example, it is possible to assemble a program and execute it immediately; likewise, a program may be compiled and executed immediately; or the newly assembled and compiled programs may be combined and executed as a single job. A master binary run tape -- from which programs may be selected and executed as required -- can also be created. As was mentioned above, Mod 1 operations may be controlled through the use of either external or internal directors (and, in many situations, it may be desirable to use a combination of both types of operating directors). Several methods of operations are described in subsequent paragraphs; in the examples shown, use is made of external directors.

Operation with Source Programs on More Than One BRT

Figure 2-2 illustrates several operations (on a job level) which are to be performed; these operations consist of:

1. Assembling programs A and D;
2. Compiling programs B and C; and
3. Executing the programs in the sequence A, B, C, D.

The Easycoder assembler is loaded from the system tape (tape 0 in this example); programs A and D are then assembled and written on tape 2. As part of the same job (or separately, if desired), the COBOL compiler is loaded from the system tape; programs B and C are then compiled and written on tape 3. Following this, programs A, B, C, and D may be executed in the specified sequence. Using the external directors shown in Figure 2-2 for program execution, the programs are sequenced automatically until the last director is read; at this point processing of the last program (D) is completed and control is returned to the operator.

For the job described above, no program file preparation was performed prior to executing the data processing function. However, source programs can be maintained in their symbolic form (as illustrated), and/or they can be maintained in their machine-language form. An important feature to note is that programs can be selected for execution directly from the output tapes of the various language processors.

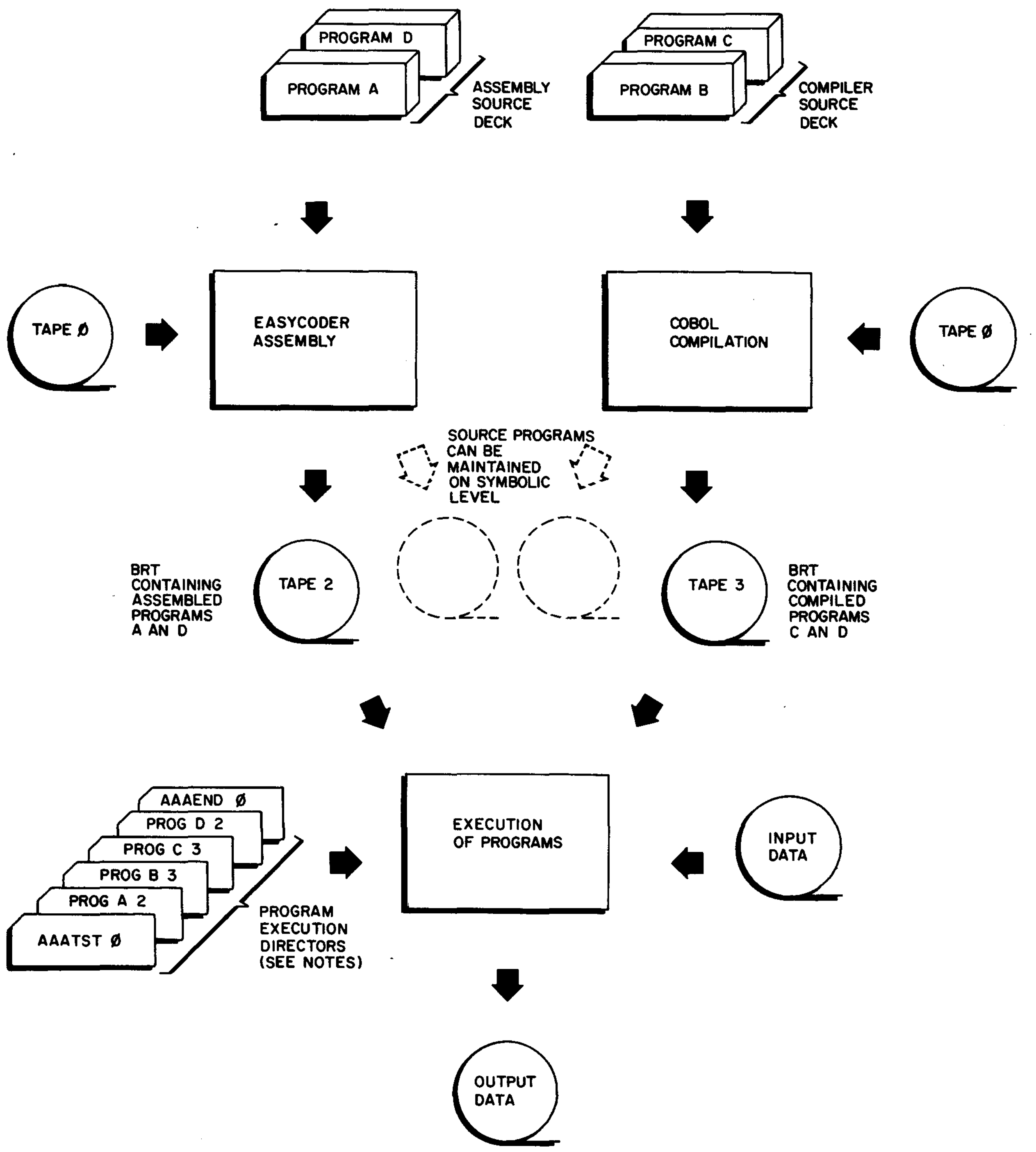
Combining Source Programs for Operation from One BRT

As previously explained, and illustrated in Figure 2-2, a program can be assembled (or compiled) and then executed directly from the BRT on which it was placed. In addition to this familiar capability, the program editing and maintenance function provides facilities which enable programs produced by the various Mod 1 language processors to be combined on a single binary run tape.

Figure 2-3 illustrates a job which consists of several parts; these include:

1. Selecting programs from output tapes produced by two different language processors;
2. Writing the selected programs in the desired order on a single BRT; and
3. Executing these programs in the designated sequence.

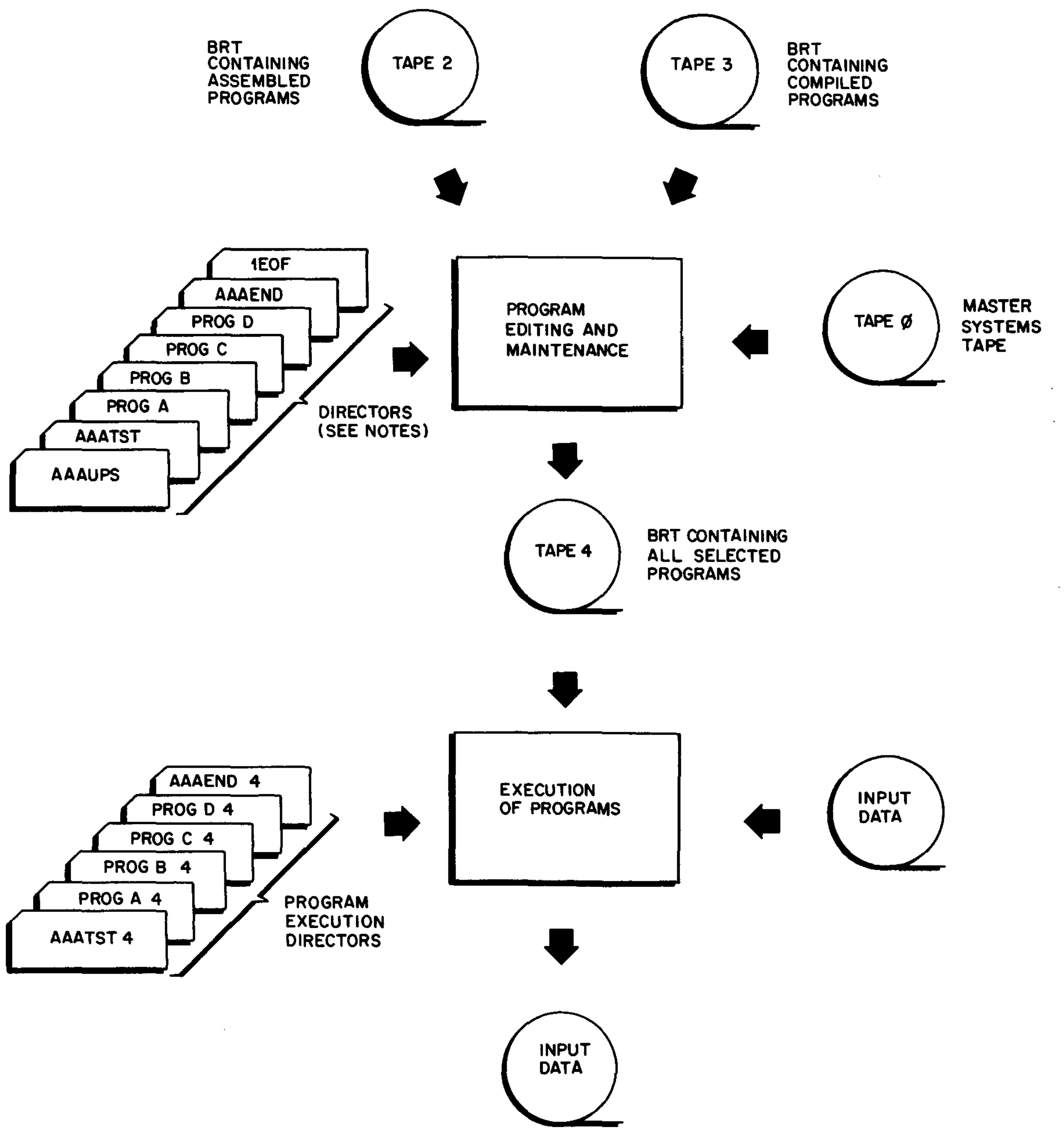
As may be interpreted from the illustration, any number of programs may have been batched for assembly or compilation. The selection process of the program editing and maintenance function provides the facility for selecting programs from various binary run tapes and subsequently writing them in any order to produce the desired program run tape or program file. The selection and ordering of programs are specified by the input directors to program editing and maintenance. In the example shown in Figure 2-3, the directors designate the inclusion of the process (AAATST) which provides automatic sequencing of the four programs A, B, C, and D during their subsequent program execution. The directors, in addition to designating the required data processing programs (A, B, C, and D in this case), designate the process (AAAEND) which properly terminates the automatic sequencing of the programs after the job is completed during subsequent program execution. Thus, the output of program editing and maintenance is a BRT containing AAATST, programs A, B, C, and D, and AAAEND. The job now proceeds to the execution of the four selected programs which perform the appropriate data processing functions that have been selected (from the single binary tape) to run as a job.



NOTES:

1. The first director (AAATST) directs the automatic sequencing of programs without halting until the end of the job is reached.
2. The next four directors cause programs A, B, C, and D (respectively) to be loaded and executed.
3. The last director (AAAEND) designates the end of the job and causes control to be returned to the operator.

Figure 2-2. Operation with Source Programs on More Than One BRT



NOTES:
1. The first director (AAAUPS) directs the loading and execution of the updating and selection processes.
2. The next six directors specify the programs to be selected from tapes 2 and 3.
3. The last director (1EOF) designates that the last input director to the updating and selection process has been reached; control is then returned to the operator.
4. For execution, the program directors direct loading from tape 4 and automatic sequencing of programs A through D; then, when the end of job has been reached, control is returned to the operator.

Figure 2-3. Combining Source Programs for Operation from One BRT

APPENDIX A

MOD 1 PUBLICATIONS

The set of publications associated with the Series 200/Operating System-Mod 1 is illustrated in Figure A-1. (Where applicable, the file number of each publication is also shown.) The publications, listed under the headings described in this manual, are summarized below.

PROGRAM PREPARATION AND MAINTENANCE

1. Language Processing Functions: The two programmers' reference manuals listed in Figure A-1, while describing the over-all operation of Series 200 hardware, provide the programmer with a detailed description of the Easycoder language elements. The COBOL and FORTRAN manuals give the programmer a detailed functional description of the respective programs. The remaining manuals supply the programmer and operator with detailed information for the various assemblers and conversion programs.
2. Program Editing and Maintenance Functions: These manuals provide both programming and operating information for those processes dealing with the storing, modifying, and maintenance of assembly-language and binary-executable programs.

PROGRAM CONTROL

1. Operation Control Functions: These publications supply both the programmer and operator with information in those elements dealing with loading, monitoring, and program sequencing.
2. Input/Output Control Functions: These publications supply programming and operating information for those processes which may be specialized and incorporated into the user's data processing system.
3. Program Test Functions: These publications supply programming and operating information for those processes which may be used either separately or as part of an automatic checkout subsystem.

UTILITY

1. Data Transcription and Editing Functions: These publications supply programming and operating information for those programs concerned with tape handling, media conversion, report generating, tabulator simulating, and data sorting and collating.
2. Mathematical Processing Functions: The manual entitled Statistics Package D provides programming information for this set of five programs that enables the user to perform various statistical analyses on numerical data.

Introduction to Series 200/Operating System-Mod 1 (258)

Operating System-Mod 1 Operating Procedures Summaries (069)

PROGRAM PREPARATION AND MAINTENANCE

LANGUAGE PROCESSING

Series 200 Programmers' Reference Manual, Models 200, 1200, 2200 (139)
Series 200 Programmers' Reference Manual, Model 120 (141) } Include Language Elements
Easycoder Assemblers C and D (041)
Transition to Easycoder — A Programmed Text (DSI-319)
Programming With Easycoder — A Programmed Text (238)
Programming With Easycoder (Level 2 Assignments) (008)
Analyzer C (019)
Library Processor C and D (051)
COBOL Compiler D (Vol. I) — A Programmed Text (083)
COBOL Compiler D (Vol. II) — A Programmed Text (091)
COBOL Compiler D and H (065)
Fortran Compiler D (027)
Fortran Conversion Techniques (002)
Fortran Compiler D Generated Object Code (003)
Easytran Symbolic Translator B and C (035)
Easytran Symbolic Translator D (220)
Easytran Program Modifier C (147)

PROGRAM EDITING AND MAINTENANCE

Update and Select C and D (025)
SPT Merge C (152)
BRT Punch C (020)
PLUS Drum Program Store (Drum Program Store C) (DSI-411)

PROGRAM CONTROL

OPERATION CONTROL

Card Loader-Monitor B (154)
Tape Loader-Monitor C (221)
Floating Tape Loader-Monitor C and Interrupt Control D (005)
Drum Bootstrap-Loader C (DSI-415)
PLUS Drum Monitor (Drum Monitor C) (DSI-408)
DIAL 200 (Drum Interrogation, Alteration, and Loader C) (DSI-404)

INPUT/OUTPUT CONTROL

1/2-Inch Tape I/OB and C (010)
1/2-Inch Tape and Terminal I/O B and C (167)
TYRO 2 (Terminal Console I/O for Easycoder C) (DSI-413)
DIPDOP (Drum I/O C) (DSI-405)

PROGRAM TEST

Program Test System C (049)
PLUS-Memory and Tape Dump Routines (Memory and Tape Dump C) (DSI-341)

UTILITY

DATA TRANSCRIPTION AND EDITING

Tape Handling Routine B (applicable to Tape Handling Routine C) (017)
Tape Sort C (V) and Collate C (V) (207)
Tape Sort C and Collate C (018)
Simultaneous Sort and Print (201)
Simultaneous Media Conversion A and C (021)
Data Conversion A and C (231)
Tape to Printer A and C (006)
Drum Sort C (157)

MATHEMATICAL PROCESSING

Statistical Package D (159)

NOTE: When ordering these publications please specify title and number. The number (enclosed in parentheses and following the title) is either an order number (e.g., 019) or a DSI number (e.g., DSI-408).

Figure A-1. Mod 1 Publications

COMPUTER-GENERATED INDEX

ASSEMBLERS, 2-4
ASSISTS
      MOD 1 ASSISTS ALL MEMBERS OF THE DATA PROCESSING
            TEAM, 1-3
BENEFITS
      MAJOR BENEFITS OF THE SERIES 200/OPERATING SYSTEM -
            MOD 1, 1-2
BRT
      COMBINING SOURCE PROGRAMS FOR OPERATION FROM ONE
            BRT, 2-16, 2-19
      OPERATION WITH SOURCE PROGRAMS ON MORE THAN ONE BRT,
            2-16, 2-18
COBOL COMPILERS, 2-4
COMBINING SOURCE PROGRAMS
    " FOR OPERATION FROM ONE BRT, 2-16, 2-19
COMPILERS
      COBOL COMPILERS, 2-4
      FORTRAN COMPILERS, 2-5
CONTROL
    " FUNCITONS,
            INPUT/OUTPUT CONTROL FUNCITONS, 2-9
    " FUNCTIONS,
            OPERATION CONTROL FUNCTIONS, 2-8
      PROGRAM CONTROL, A-1, 2-8
    " PROGRAMS,
            PROGRAM CONTROL PROGRAMS, 2-10
CONVERSION TRANSLATORS, 2-5
DATA
    " PROCESSING TEAM,
            MOD 1 ASSISTS ALL MEMBERS OF THE DATA PROCESSING
                  TEAM, 1-3
    " TRANSCRIPTION AND EDITING FUNCTIONS, 2-13
EDITING
    " FUNCTIONS,
            DATA TRANSCRIPTION AND EDITING FUNCTIONS, 2-13
      PROGRAM EDITING AND MAINTENANCE FUNCTIONS, 2-6
EXPERIENCE
      OPERATING SYSTEM EXPERIENCE,
            HONEYWELL'S OPERATING SYSTEM EXPERIENCE, 1-1
FLEXIBILITY AND ORDERLY GROWTH POTENTIAL ARE PROVIDED, 1-3
FORTRAN COMPILERS, 2-5
FUNCITONS
      INPUT/OUTPUT CONTROL FUNCITONS, 2-9
FUNCTIONS
      EDITING FUNCTIONS,
            DATA TRANSCRIPTION AND EDITING FUNCTIONS, 2-13
      LANGUAGE PROCESSING FUNCTIONS, 2-2
      MAINTENANCE FUNCTIONS,
            PROGRAM EDITING AND MAINTENANCE FUNCTIONS, 2-6
      MATHEMATICAL PROCESSING FUNCTIONS, 2-14
      MOD 1 FUNCTIONS, 2-3
      OPERATION CONTROL FUNCTIONS, 2-8
      PROGRAM TEST FUNCTIONS, 2-11
      SYSTEM FUNCTIONS, 2-2
GROWTH POTENTIAL
      FLEXIBILITY AND ORDERLY GROWTH POTENTIAL ARE
            PROVIDED, 1-3
INPUT/OUTPUT CONTROL FUNCITONS, 2-9
INSTALLATION MANAGER, 1-3
INTRODUCTION, 1-1
LANGUAGE PROCESSING FUNCTIONS, 2-2
MAINTENANCE
    " FUNCTIONS,
            PROGRAM EDITING AND MAINTENANCE FUNCTIONS, 2-6
      PROGRAM PREPARATION AND MAINTENANCE, A-1, 2-7
      PROGRAM PREPARATIONS AND MAINTENANCE, 2-2
MAJOR BENEFITS OF THE SERIES 200/OPERATING SYSTEM -
            MOD 1, 1-2
MANAGER
      INSTALLATION MANAGER, 1-3
MATHEMATICAL PROCESSING FUNCTIONS, 2-14
MAXIMIZED
      THROUGHPUT IS MAXIMIZED, 1-2
MAXIMUM USE IS MADE OF MEMORY AND PERIPHERAL UNITS, 1-3
MEMBERS
      MOD 1 ASSISTS ALL MEMBERS OF THE DATA PROCESSING
            TEAM, 1-3
MEMORY
      MAXIMUM USE IS MADE OF MEMORY AND PERIPHERAL UNITS,
            1-3
MINIMIZED
      OVERHEAD IS MINIMIZED, 1-3
      TURNAROUND TIME IS MINIMIZED, 1-2
MOD
      MAJOR BENEFITS OF THE SERIES 200/OPERATING SYSTEM -
            MOD 1, 1-2
      SERIES 200/OPERATING SYSTEM - MOD 1, 1-4, 2-1
    " 1 ASSISTS ALL MEMBERS OF THE DATA PROCESSING TEAM,
            1-3
    " 1 FUNCTIONS, 2-3
    " 1 PUBLICATIONS, A-1, A-2
OPERATING SYSTEM EXPERIENCE
      HONEYWELL'S OPERATING SYSTEM EXPERIENCE, 1-1
OPERATION
      COMBINING SOURCE PROGRAMS FOR OPERATION FROM ONE
            BRT, 2-16, 2-19
    " CONTROL FUNCTIONS, 2-8
      SYSTEM OPERATION, 2-15
    " WITH SOURCE PROGRAMS ON MORE THAN ONE BRT, 2-16,
            2-18
OPERATOR, 1-3
ORDERLY GROWTH POTENTIAL
      FLEXIBILITY AND ORDERLY GROWTH POTENTIAL ARE
            PROVIDED, 1-3
OVERHEAD IS MINIMIZED, 1-3
PERIPHERAL UNITS
      MAXIMUM USE IS MADE OF MEMORY AND PERIPHERAL UNITS,
            1-3
POTENTIAL
      ORDERLY GROWTH POTENTIAL,
            FLEXIBILITY AND ORDERLY GROWTH POTENTIAL ARE
                  PROVIDED, 1-3
PREPARATION
      PROGRAM PREPARATION AND MAINTENANCE, A-1, 2-7
      PROGRAM PREPARATIONS AND MAINTENANCE, 2-2
PROCESSING
    " FUNCTIONS,
            LANGUAGE PROCESSING FUNCTIONS, 2-2
            MATHEMATICAL PROCESSING FUNCTIONS, 2-14
    " STRUCTURE, 2-1
    " TEAM,
            MOD 1 ASSISTS ALL MEMBERS OF THE DATA PROCESSING
                  TEAM, 1-3
PROGRAM
    " EDITING AND MAINTENANCE FUNCTIONS, 2-6
    " PREPARATION,
            PROGRAM PREPARATION AND MAINTENANCE, A-1, 2-7
            PROGRAM PREPARATIONS AND MAINTENANCE, 2-2
    " TEST FUNCTIONS, 2-11
PROGRAM CONTROL, A-1, 2-8
    " PROGRAMS, 2-10
PROGRAMMER, 1-3
PROGRAMS
      COMBINING SOURCE PROGRAMS FOR OPERATION FROM ONE
            BRT, 2-16, 2-19
      PROGRAM CONTROL PROGRAMS, 2-10
      SOURCE PROGRAMS,
            OPERATION WITH SOURCE PROGRAMS ON MORE THAN ONF
                  BRT, 2-16, 2-18
      UTILITY PROGRAMS, 2-14
PUBLICATIONS
      MOD 1 PUBLICATIONS, A-1, A-2
SERIES 200/OPERATING SYSTEM
    " - MOD 1, 1-4, 2-1
      MAJOR BENEFITS OF THE SERIES 200/OPERATING SYSTEM -
            MOD 1, 1-2
SOURCE PROGRAMS
      COMBINING SOURCE PROGRAMS FOR OPERATION FROM ONE
            BRT, 2-16, 2-19
      OPERATION WITH SOURCE PROGRAMS ON MORE THAN ONE BRT,
            2-16, 2-18
STRUCTURE
      PROCESSING STRUCTURE, 2-1
SYSTEM
    " EXPERIENCE,
            HONEYWELL'S OPERATING SYSTEM EXPERIENCE, 1-1
    " FUNCTIONS, 2-2
    " OPERATION, 2-15
      SERIES 200/OPERATING SYSTEM,
            MAJOR BENEFITS OF THE SERIES 200/OPERATING
                  SYSTEM - MOD 1, 1-2
      SERIES 200/OPERATING SYSTEM - MOD 1, 1-4, 2-1
TEAM
      DATA PROCESSING TEAM,
            MOD 1 ASSISTS ALL MEMBERS OF THE DATA PROCESSING
                  TEAM, 1-3
TEST FUNCTIONS
      PROGRAM TEST FUNCTIONS, 2-11
THROUGHPUT IS MAXIMIZED, 1-2
TIME
      TURNAROUND TIME IS MINIMIZED, 1-2
TRANSCRIPTION
      DATA TRANSCRIPTION AND EDITING FUNCTIONS, 2-13
TRANSLATORS
      CONVERSION TRANSLATORS, 2-5
TURNAROUND TIME IS MINIMIZED, 1-2
UNITS
      PERIPHERAL UNITS,
            MAXIMUM USE IS MADE OF MEMORY AND PERIPHERAL
                  UNITS, 1-3
UTILITY, A-1, 2-13
    " PROGRAMS, 2-14
200/OPERATING SYSTEM
      MAJOR BENEFITS OF THE SERIES 200/OPERATING SYSTEM -
            MOD 1, 1-2
      SERIES 200/OPERATING SYSTEM - MOD 1, 1-4, 2-1

HONEYWELL EDP TECHNICAL PUBLICATIONS
USERS' REMARKS FORM

TITLE: SERIES 200 INTRODUCTION TO SERIES 200/OPERATING SYSTEM-MOD 1 SOFTWARE BULLETIN	DATED: MARCH, 1966
	FILE NO: 122.0005.001C.0-258

ERRORS NOTED:

Fold

Cut Along Line

SUGGESTIONS FOR IMPROVEMENT:

Fold

FROM: NAME ______________________ DATE ____________

COMPANY ______________________

TITLE ______________________

ADDRESS ______________________

____

FIRST CLASS
PERMIT NO. 39531
WELLESLEY HILLS
MASS.

Cut Along Line

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

POSTAGE WILL BE PAID BY

HONEYWELL
ELECTRONIC DATA PROCESSING DIVISION
60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATT'N: TECHNICAL COMMUNICATIONS DEPARTMENT

Honeywell
ELECTRONIC DATA PROCESSING