3400
3600
3800

COMPUTER SYSTEMS
COMPASS

TRAINING MANUAL
VOLUME III

RECORD of REVISIONS

REVISION	NOTES
A (3-1-66)	Final Edition Released.
B (4-19-67)	Publications Change Order CA16274. Manual enlarged from one to three volumes, replacing 3600 COMPASS Programming Guide, Pub. No. 60166700, which becomes Volume 2 of the new 3-volume set. Miscellaneous corrections to Volume 2, pages 1-1, 1-21, 1-26, 2-10, 2-14, 7-3, 7-10, 11-5, 12-26, 12-30, 13-25, 15-13, 15-18, 15-22, 15-30, 15-42, 16-6, 17-2, 17-3, 17-6, 18-2, 18-6, 18-10, 18-14, 18-18, and 18-22. This edition obsoletes all previous editions.

FORM CA230 REV. 1-67

Pub No. 60166700

Printed in United States of America

Address comments concerning this manual to:

Control Data Corporation
Marketing Training Department
8100 - 34th Avenue South
Minneapolis, Minnesota 55440

FOREWORD

This manual is intended as a guide in learning how to program the upper 3000 computer systems. It includes a hardware concept of the systems, the use of the COMPASS programming language, and the use of the SCOPE monitor. Step-by-step example problems, with and without given solutions, are included to develop the capability of using the language.

This manual is a major revision to and a replacement for the 3600 Computer System COMPASS Programming Guide and retains the same publication number. It is now expanded to three volumes.

Volume I

This volume consists of three sections. The first section deals with the introduction to the systems. The second section deals with the central processor. The third section deals with problem-oriented exercises in which random instructions are picked to solve problems.

Volume II

This volume consists of one section. The instruction repertoire is divided into groups. Groups 1-18 are hardware instruction groups, and groups 19-25 are pseudo instruction groups. Each group is followed by explanations of new concepts and problems designed to use instructions from the group.

Volume III

This volume consists of two sections. The first section deals with the SCOPE system. It shows how to run jobs under the system and explains new concepts such as overlay processing and library preparation. The second section contains several computer output listings obtained as a result of running the example problems under SCOPE.

REFERENCES

3400 SCOPE / COMPASS Reference Manual	Pub. No. 60057800
3400/3600/3800 Instant TAPE SCOPE	Pub. No. 60059000
3600 Computer System Reference Manual	Pub. No. 60021300
3600 COMPASS Reference Manual	Pub. No. 60052500
3600 Instant COMPASS	Pub. No. 60056500
SCOPE Reference Manual	Pub. No. 60053300
3000 Series Peripheral Equipment Reference Manual	Pub. No. 60108800

CONTENTS

VOLUME III

SECTION I

SYSTEM SOFTWARE

THE SCOPE SYSTEM

SCOPE stands for Supervisory Control Of Program Execution. It is frequently referred to as the Monitor of the upper 3000 systems. The Scope Monitor provides a system of operator and programmer aids to simplify the operator's job and increase through-put. By having a monitor for the system, operator errors, operator intervention, and amount of computer time are minimized.

Other features of the monitor include job stacking, equipment allocation, recovery and debugging aids, interrupt processing, overlay processing, and library preparation. Many of these features are explained as we go along.

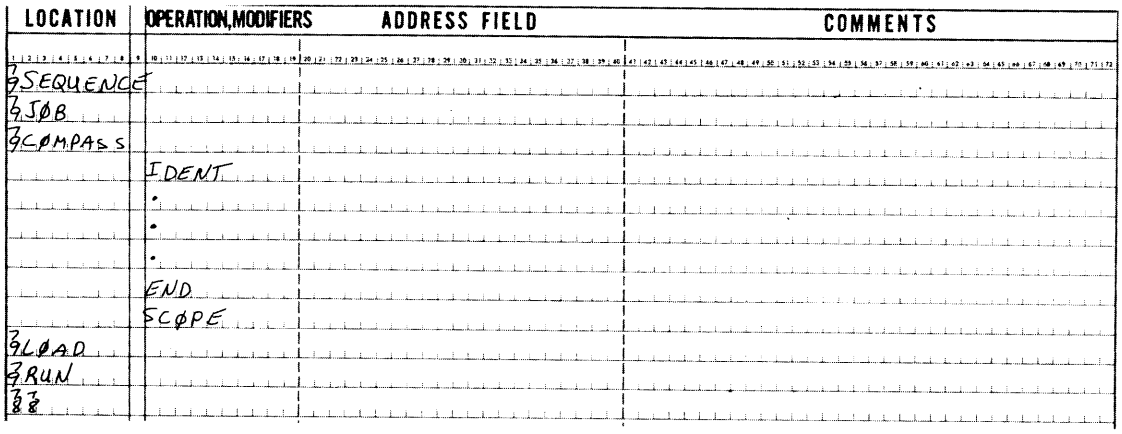
Monitor operations are specified in a job by use of control statements and programming requests. Control statements have the following card format: a 7, 9 punch in column 1 and a statement name beginning in column 2 with parameters separated by commas following. Control statements are free-field, but must be contained on a single 80 column card. No terminating character is needed. Programming requests are written as system macros and are assembled into a calling sequence to a Scope routine. This will be explained in detail later.

A job includes all operations indicated between JOB cards, or, if one job, between the JOB card and ENDSCOPE card. Each job is terminated with an end-of-file card. End-of-file cards are also used to separate runs within a job. If more than one job is to be processed, jobs are placed together (job stacking) with the ENDSCOPE card at the end of the last job. If jobs are stacked, SEQUENCE cards with sequence numbers are inserted just before each JOB card so that, if the stack was on tape, the nth job could be read, compiled, and executed bypassing the others.

Within a job other control cards determine what the monitor is to do. A COMPASS card directs the monitor to read the Compass assembler from the library into core storage.

A LOAD card directs the loader to load the binary object program into core storage. A RUN card directs the monitor to execute the object program.

In general, then, a job might look like this:



This constitutes a Compass assembly in which the object program is loaded and executed. If the program is to read and process data cards, the data cards would be inserted between the RUN card and the end-of-file card $\left(\begin{smallmatrix}7\\8\end{smallmatrix}\right)$.

UNIT SPECIFICATION

Before each statement is taken separately, it is necessary to understand how units are referenced. Units are divided into two groups - PHYSICAL and LOGICAL. Physical units have been discussed up to this point. A programmer references these units by specifying the proper data channel number, the proper controller number, and the proper unit number. If the specified unit is down (not operable), he must attempt another. This method requires that the programmer know the hardware configuration and be well versed in the hardware use of the system. To alleviate the problem a method has been devised to help the programmer program units without having to know the intimate intricacies of the hardware system. Instead of referencing physical units, he references logical units.

Logical units are specified by numbers ranging from 1-80. These units are independent of the physical units in the system. No channel, controller, or hardware designate is

necessary. Scope automatically equates a logical unit to a physical unit in the system - one which is up (operable). If the same logical unit is specified later in the program (e. g. to read back data), the same equated physical unit will be referenced.

The logical units are composed of three classes:

1.	Programmer Units	1 - 49
2.	Scratch Units	50 - 59
3.	System Units	60 - 80

Programmer Units

The programmer units are assigned throughout the job for reference by the program. The programmer may choose any number within the range. He must use the same number if he wishes to re-reference the same unit. These units can be saved at the end of a job and they will be available for reuse in a later job.

Scratch Units

The scratch units may be referenced at any time by the programmer. However, they are released after each execution and may not be saved.

System Units

The system units are assigned by the monitor system. They can be used by the monitor or by the programmer. The definitions of the system units are now given.

The Standard Input unit is logical unit 60. This unit contains the job stack. Normally it is assumed to be magnetic tape, but it may be a card reader. Card images are represented on tape as either binary or BCD records. Each control card (binary) is recorded as one binary record on tape. Each binary record contains 160 frames. Each program card (BCD) is recorded as one BCD record on tape. Each BCD record contains 80 frames. Data cards may be either binary or BCD.

The Standard Output unit is logical unit 61. This unit contains the listable output such as the source deck, memory map, and program generated answers. Normally it is assumed to be magnetic tape, but it may be the line printer. Output is recorded as 120 characters per record. Recording format is always BCD.

The Standard Punch unit is logical unit 62. This unit contains the binary object deck. Normally it is assumed to be magnetic tape, but it may be a card punch. Card images are recorded on tape as binary records each card forming 160 frames of information.

The Standard Input Comments unit and Standard Output Comments unit are logical units 63 and 64 respectively. Normally they are assumed to be the typewriter. These units allow the operator to communicate with the system through interrupt control.

The Standard Accounting unit is logical unit 65. This unit contains the job statements and the time used by each job. This can be used for billing purposes to users of the system.

The Standard Load-and-Go unit is logical unit 69. This unit contains binary object programs. If an execution is to be performed, the information is loaded and run. This unit is normally magnetic tape. Output is recorded in binary format with 160 frames to each record. The writing of this tape usually means an execution of the program is desired.

The Standard Library unit is logical unit 70. This unit contains the Scope library. It includes the Scope Monitor, the Fortran compiler, the Cobol compiler, the Compass assembler, and the object time routines, just to name a few.

Logical units 66, 67 and 68 are Scope reserved units. Any attempt to reference these units will cause the job to terminate abnormally.

Logical units 71-79 are auxiliary library units. These are magnetic tapes and are prepared using PRELIB. More about this is given at the end of this section.

SCOPE CONTROL STATEMENTS

Scope controls each job as it is processed. Through the use of control cards the programmer can see to it that the job is monitored during compilation and execution. Safeguards are taken so that the computer is not caught in an endless loop or that an output tape is not destroyed by a following job.

In order for Scope to perform the right operation at the right time it is important for the programmer to follow the proper field designations per control card and to see to it that the control cards are in the proper order.

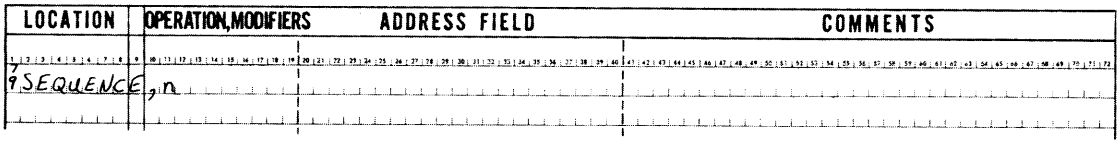
A control card is a binary card which has a 7, 9 punch in column 1. Beginning in column 2 the statement name is given followed by parameters separated by commas.

The first set of control cards given are:

1. SEQUENCE
2. JOB
3. COMPASS
4. SCOPE
5. LOAD
6. RUN
7. END-OF-FILE

The SEQUENCE Control Statement

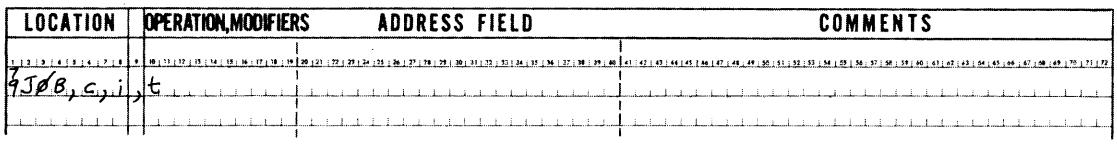
FORM:



The SEQUENCE control statement assigns a job sequence number, n, to the following job. When this statement is encountered, saved tapes, programmer units, scratch units, and the load-and-go unit are released. Released tapes that are not unloaded are available to the next job.

The JOB Control Statement

FORM:



All programs submitted for processing under Scope start with a JOB statement which signals the beginning of a job, provides accounting information for the installation, identifies the programmer, and sets a job processing time limit.

The parameters of the statement are defined as follows:

c	the charge number:	it may be an unlimited number of alpha-numeric characters.
i	the programmer identification:	it may be any length and appears as given in the control card listing; it is truncated to 6 characters for operation identification or tape labels.
t	the maximum time limit:	the maximum in minutes allowed for the entire job including operator functions. No job may exceed 2236 minutes; if blank, the maximum time is assumed.

The job is terminated if the c and i fields are not present. A single JOB card may be used for any number of independent programs; however, the time specified is the maximum allowed for the combined programs.

be produced on unit 69 (standard load-and-go).

L = u List option where u may assume values 1-49 or 61; if the parameter is absent, no listing will be produced. If only L appears, the listing will appear on unit 61 (standard output).

Y = u Cosy input medium where the logical unit u may assume values 1-49 or 60; if the parameter is absent, input from unit 60 (standard input) is assumed.

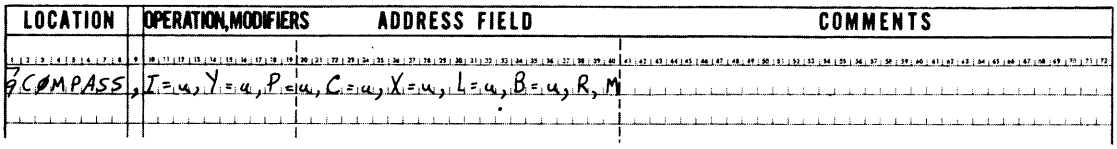
B = u Hollerith output where u may assume values 1-49 or 62; logical unit u must be specified. If the C option is specified, the B option is ignored.

R List cross reference symbol table on unit 61 (standard output).

M List programmer macros. The source card images of the programmer macros are listed when the macros are called. The source card images of generative coding is also listed.

The COMPASS Control Statement

FORM:



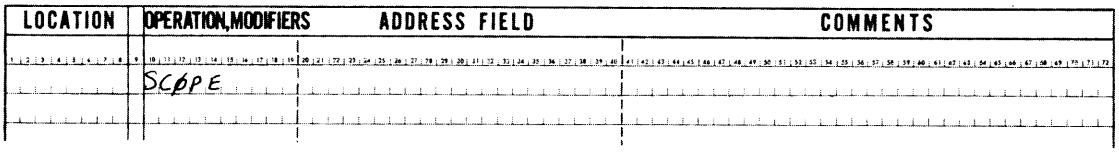
The COMPASS statement is one of several program name statements representing compilers or assemblers. Other examples that are included in this category are; FORTRAN, COBOL and ALGOL. The purpose of a program name statement is to call the compiler or assembler from the library tape, read it into core, and turn control over to it. The cards following the program name statement must be in the format which is unique to its language.

Column 1 of the control card contains punches in rows 7 and 9. Starting in column 2, the word COMPASS appears, followed by a comma and up to 9 free field parameters separated by commas. The 9 parameters are options that may have different forms or are left out entirely. The 9 options are;

- I = u Hollerith input medium where the logical unit u may assume values 1-49 or 60; if the parameter is absent, input from unit 60 (standard input) is assumed.
- P = u Punch option where the logical unit u may assume values 1-49 or 62; if the parameter is absent, no binary output will be produced on unit 62 (standard punch).
- C = u Cosy output option where the logical unit u may assume values 1-49 or 62; if the parameter is absent, no Cosy output will be produced. If only C appears, Cosy output will be produced on unit 62 (standard punch).
- X = u Binary output for load-and-go unit where u may assume values 1-49 or 69; if the parameter is absent, no load-and-go tape will be written. If only X appears, binary output for load-and-go will

The SCOPE Statement

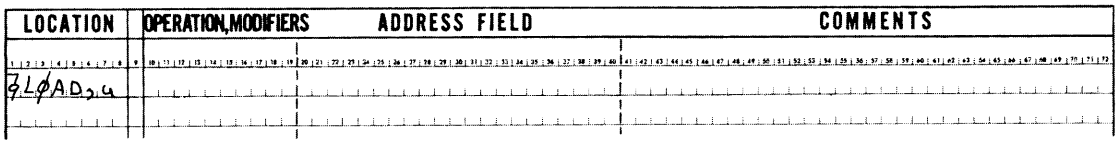
FORM:



The SCOPE statement terminates the assembly or compilation process and causes the return of control to the Scope Monitor. The SCOPE card begins in column 10. This statement is not really a Scope control statement in a true sense. This statement is actually read by the assembler.

The LOAD Control Statement

FORM:

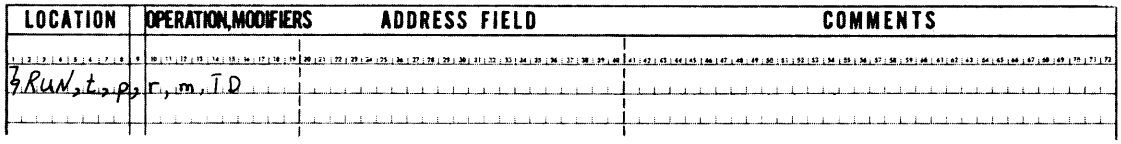


The LOAD statement will cause the loader to load relocatable binary subprograms into storage from programmer units or the load-and-go unit. The unit u is a logical unit number having values 1-49 or 69. If u is omitted (also comma), the standard load-and-go unit (69) is assumed.

When the LOAD statement is encountered, Scope backspaces unit u one file and loads subprograms until an end-of-file, two transfer cards, or another control card is encountered. If the unit cannot be backspaced, Scope immediately loads the subprograms. Scope interprets a second transfer card as a loader terminator, but it is not required. If binary subprograms, transferred from the standard input unit and produced by compilation or assembly, are stored on the same logical unit during the job, only one end-of-file mark will be present and it will follow the last subprogram stored on the unit.

The RUN Control Statement

FORM:



The RUN statement initiates program execution by transferring control to the object program in sotrage. This statement is required to execute all object programs.

The parameters are defined as follows:

t — the execution time limit in minutes (maximum 2236). The entire job is terminated if the limit is exceeded. If t is blank, a constant time limit, determined by the installation, is used. If the run time limit is greater than the remaining job time limit, execution continues only until the job time is depleted. The run limit may not equal zero.

p — the maximum number of print or write operations which may be requested on the standard output unit during the execution. This includes debugging dumps and any other output during execution. The entire job is terminated if the print limit is exceeded. If the print limit is blank, a constant print limit, determined by each installation, is used. The print limit may not equal zero.

r — the recovery indicator specifies an area to be dumped if the program does not proceed to normal completion. The recovery dump is written on standard output (unit 61) and uses the following indicators:

r	dumped area
0 or blank	console
1	program and console
2	labeled common and console
3	program, labeled common, and console
4	numbered common and console
5	program, numbered common, and console
6	labeled common, numbered common, and console
7	complete area used except resident Scope

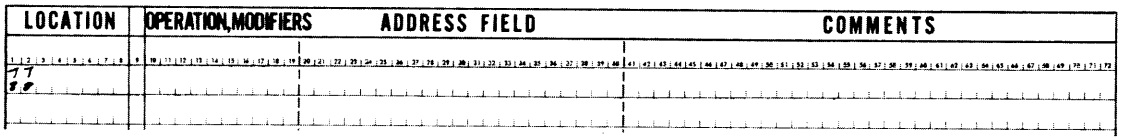
m the memory map indicator. If m is blank, storage allocations after loading will be listed on the standard output unit. No map is written if m is any other character.

d dump indicator. A dump is to be taken after normal termination of the program if d is not blank.

A RUN statement is used after a LOAD statement or after a binary program on standard input (unit 60).

The END-OF-FILE Statement

FORM:

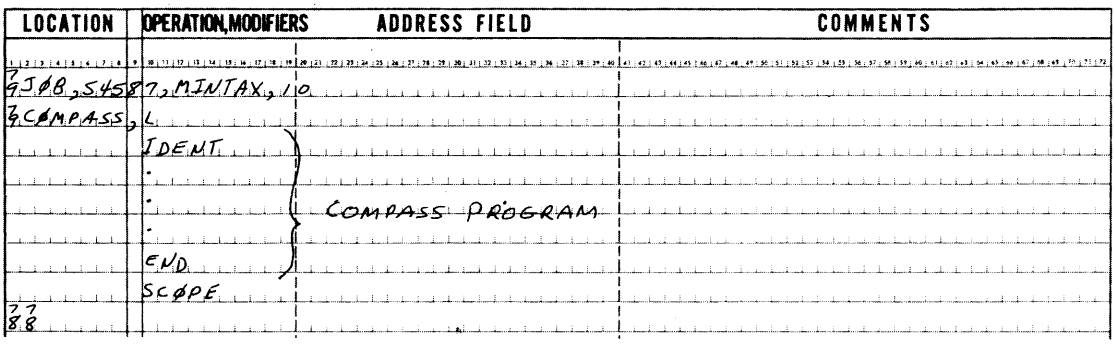


The END-OF-FILE statement is required to separate jobs and is frequently used to separate runs within a job. A 7, 8 multiple punch represents a 17_8 code in BCD. This is an end-of-file mark on tape.

If fatal errors occur during assembly or compilation of a program, loading is not attempted. Subsequent assemblies or compilations of programs in the job are processed, however, if the program name control statement is preceded by an end-of-file.

EXAMPLES OF DECK STRUCTURE

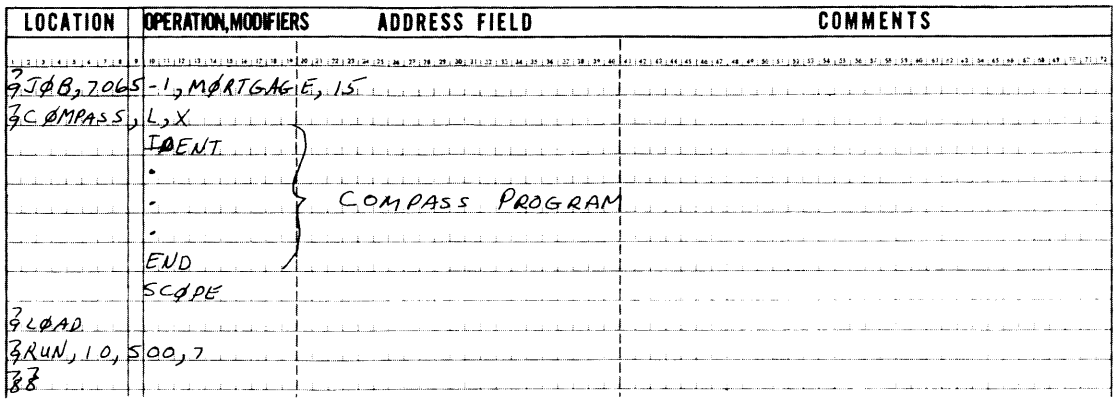
1) Assembly of a COMPASS routine



In this job the programmer is interested in obtaining a listing of the source program with any assembly errors.

The job is called MINTAX, has a control number of 54587, and will assemble for a maximum of 10 minutes. When the COMPASS card is read by Scope, the CARD is scanned to the first comma. Recognizing it as a COMPASS control card, Scope will read the Compass assembler into core and turn control over to it (exit Scope - enter Compass). Compass then scans, interprets, and assembles all cards until the Scope card is read. When it is scanned, control is turned back to Scope (exit Compass - enter Scope), which reads the next card. The end-of-file is read by Scope. The card after the end-of-file might be another JOB card if jobs are stacked.

2) Assembly and execution of a COMPASS routine



In this job the programmer is interested in obtaining a listing of the source program with any assembly errors. He also wishes execution if no assembly errors occur.

The job is called MORTGAGE, has a control number of 7065-1, and has a maximum time of 15 minutes for assembly and execution or 10 minutes for execution. The Scope Monitor reads the JOB card and processes the parameters. Upon encountering the END card, the source program is listed on unit 61 with errors, if any. The object program (relocatable binary) formed from the source program is then recorded on unit 69.

The next card read is SCOPE This card transfers control back to the monitor, which reads the next card. The LOAD card directs the loader to read unit 69 (since no unit specified) into core destroying the assembler (not needed anymore).

When the RUN card is read, Scope extracts the parameters. The execution time limit is extracted and compared with the remaining time left from the limit on the JOB card. Whichever is less at this point is the effective time limit used for the remainder of the job. The maximum print limit is recorded and the recovery indicator is set. Program control then goes to the transfer address of the loaded program and execution begins.

When the program is finished (assuming no errors), control goes back to Scope which reads the end-of-file, and the job is finished.

3) Assembly and execution of a COMPASS routine with Cosy output.

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	7JOB,.8816581.,DJE,6		
	7COMPASS,	L,X=10,C=20	
		IDENT	
		.	
		.	
		.	
		END	
		SCOPE	
	7LOAD,10		
	7RUN,S,30000,1		
		.	
		(Data cards)	
		.	
	88		

This job consists of an assembly and execution of a Compass program having data cards to be processed during the execution portion. The data cards are within the job and are read from standard input (unit 60).

The programmer is interested in obtaining a listing of the source cards and an output of source in Cosy format. The Cosy output goes to logical unit 20 and can be saved (shown later) for a later job.

The relocatable binary formed from the assembly of the program is output to unit 10. When the assembler transfers control back to the monitor through the use of the SCOPE card, the monitor loads unit 10 into core. The RUN card is read by the monitor and execution is initiated.

The data cards are next on standard input and they are read from unit 60 by the Compass program, processed, and the results output on unit 61. The data cards are read up to but not including the end-of-file. When the program is finished, control is given back to Scope. Scope reads the end-of-file and goes on to the next job.

4) Execution of a job with no assembly.

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
7JOB,9001,	MISLED,3.		
	(binary object program)		
7RUN,3,300,1.			
88			

In this case the binary object program (a relocatable binary deck) is inserted in the job. It has the same form as was output to the load-and-go unit during the assembly of Compass programs in the previous example.

Scope reads the JOB card and records the parameters. Upon encountering the relocatable binary deck, the Scope loader immediately loads it into core from standard input. The RUN card is read, parameters processed, and execution of the program begins. When the program is finished, control is transferred to Scope which reads the end-of-file.

5) Assembly and execution of multi-jobs.

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	7		
	9SEQUENCE,1		
	7		
	9JØB,50000,MAX,5		
	7		
	9CØMPASS,L,P,X		
	IOENT		
	.		
	.		
	.		
	END		
	SCØPE		
	7		
	9LØAD		
	7		
	9RUN,5,100,1		
	7		
	8		
	7		
	9SEQUENCE,2		
	7		
	9JØB,DEPT-4,MIN,10		
	.		
	(binary object program)		
	.		
	7		
	9RUN,10,3500,1		
	7		
	8		

These jobs processed together are the same type used previously except the SEQUENCE card with its number is shown. SEQUENCE cards are usually given when there is more than one job to be processed. SEQUENCE cards go just before the job that they represent.

SYSTEM MACROS

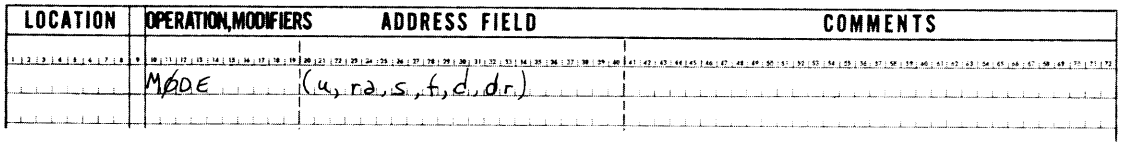
In the previous section we discussed programmer macros. We found that programmer macros are defined by the programmer at the beginning of the subprogram, and then are called at various places within the subprogram. Each time they are called the macros are assembled with possible parameter substitutions.

At this time we introduce another type of macro - the SYSTEM MACRO. The system macro uses the same idea as used with the programmer macro, except that the system macro is already defined. A system macro exists on the library tape and can be called by a calling sequence in the form of a macro. The standard definition of a macro is set, and any programmer wishing to incorporate (call) the macro into his program must know and follow the format of the macro call precisely in order to have it inserted into his program.

Before we discuss how the system macros can be incorporated and used in a Compass subprogram, let's first define what they are and how they're formatted. The first half will include the I/O requests and the second half will include all other requests.

The MODE Request

FORM:



The MODE request defines the operating or recording mode of a tape unit. The request will only be performed if the unit is available for use. If available, the request will be honored and program control will continue to the next Compass instruction. If unavailable, program control will go to the reject address. In effect, this request is similar to the hardware EXTF instruction. The parameters for this instruction are defined as follows:

	Parameter	Meaning
1.	u	Logical unit number
2.	ra	Reject address
3.	s	Specifies an operating condition for the unit. The allowable parameters are; a. RW (read and write) - all legal requests will be performed. b. BY (bypass) - all requests except STATUS or MODE will be treated as no operation until the end of the job.

c. RO (read only) - WRITE, WRLABEL, WEOT, MARKEF, or ERASE requests will be rejected.

4. f — Specifies the format of the unit. The allowable parameters are;

a. BCD - Binary coded-decimal

b. BIN - Binary

5. d — Density. The allowable parameters are;

a. HY - hyper density (800)

b. HI - high density (556)

c. LO - low density (200)

d. OP - operator, use density of mounted tape.

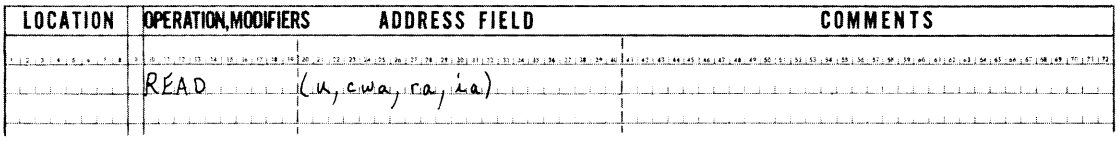
6. dr — Direction of tape. If not specified, normal is assumed. The allowable parameters are;

a. ND - normal direction for READ or BSPR requests.

b. RV - reverse direction for the READ or BSPR requests. Data is stored according to the control word specification with no alteration.

The READ Request

FORM:

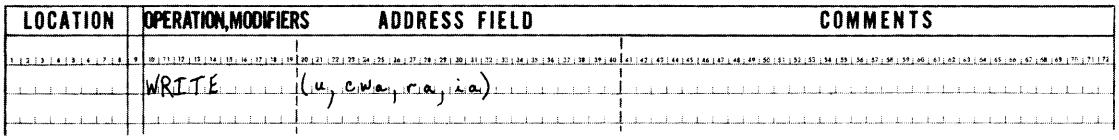


The READ request activates a buffered read from logical unit u. If the unit is available, the read is initiated and program control goes to the next Compass instruction. If the unit is unavailable, program control transfers to the reject address.

When the read operation is completed or an abnormal condition occurs, control transfers to the interrupt subroutine at the interrupt address. If no interrupt address is specified, no interrupt will take place to the interrupt subroutine.

The WRITE Request

FORM:



The WRITE request activates a buffered write on logical unit u. If the unit is available, the write is initiated and program control goes to the next Compass instruction. If the unit is unavailable, program control transfers to the reject address.

When the write operation is completed or an abnormal condition occurs, control transfers to the interrupt subroutine at the interrupt address. If no interrupt address is specified, no interrupt will take place to the interrupt subroutine.

The REOT Request

FORM:

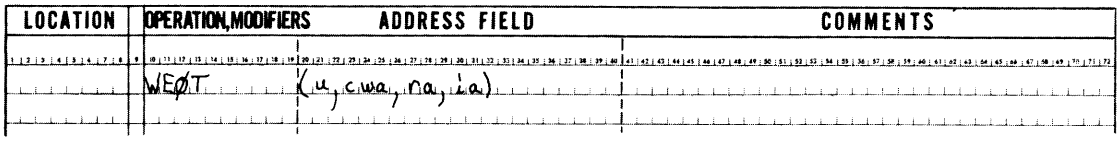
LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	REØT	(u, cwa, ra, ia)	

The REOT request is a tape movement control which allows the programmer to read after the physical end-of-tape. The tape is read from logical unit u according to the control word at the control word address. If the unit is available, the read is initiated and program control goes to the next instruction. If the unit is unavailable, program control transfers to the reject address.

When the read operation is completed or an abnormal condition occurs, control transfers to the interrupt subroutine at the interrupt address. If no interrupt address is specified, no interrupt will occur.

The WEOT Request

FORM:

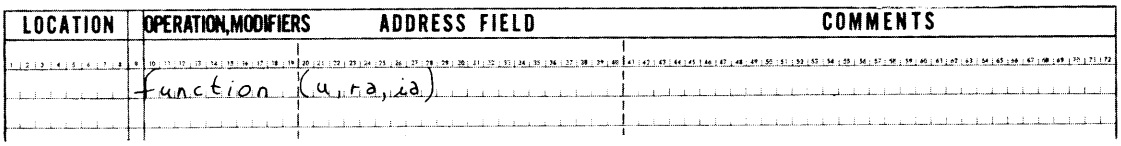


The WEOT request is a tape movement control which allows the programmer to write after the physical end-of-tape. The tape is written on logical unit u according to the control word at the control word address. If the unit is available, the write is initiated and program control goes to the next instruction. If the unit is unavailable, program control transfers to the reject address.

When the write operation is completed or an abnormal condition occurs, control transfers to the interrupt subroutine at the interrupt address. If no interrupt address is specified, no interrupt will occur.

The TAPE CONTROL Requests

FORM:



The tape control requests move tape in the forward or backward direction without transmitting data. If logical unit u can perform the request, it is initiated and program control goes to the next instruction. If the unit is unavailable, program control goes to the reject address. If the interrupt address is specified, program control goes to the interrupt routine when the request has been performed.

The tape control requests may be any of the following:

1. BSPR
2. BSPF
3. SKIP
4. MARKEF
5. ERASE
6. REWIND

1. The BSPR request backspaces the unit one record. If the unit is unassigned, this request is bypassed.

2. The BSPF request backspaces the unit one file. If the unit is unassigned, this request is bypassed.

3. The SKIP request skips to an end-of-file or end-of-tape.

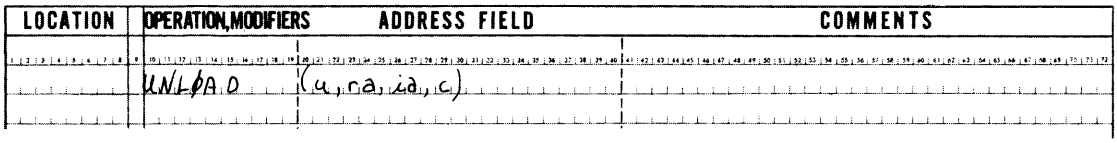
4. The MARKEF request marks an end-of-file.

5. The ERASE request erases approximately six inches of tape.

6. The REWIND request rewinds the tape to load point.

The UNLOAD Request

FORM:



The UNLOAD request causes tape unit u to unload. If the unit is unavailable, program control transfers to the reject address. If the interrupt address is specified, the computer interrupts to that address at the end of the operation.

The release code c specifies the disposition of the unit assignment after it has been unloaded:

0	unit assignment released
non-zero	unit assignment not released

If an interrupt address is specified, the unit is assumed as not released.

The RELEASE Request

FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	RELEASE	(u, na, c)	

The RELEASE request releases the assignment of logical unit u and directs the disposition of the current physical unit. If the unit is unavailable, program control transfers to the reject address.

The release code c specifies the disposition of the physical unit currently assigned to the logical unit:

0	dispose of physical unit according to previous declaration; if none is given, the tape is rewound.
non-zero	rewind the physical unit and release the assignment, but do not dispose of the tape.

The RDLABEL Request

FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	RDLABEL	(u, la, ra, ia)	

The RDLABEL request reads the label from logical unit u and transfers the label to memory starting at the label address. If the unit is unavailable, program control transfers to the reject address. If specified, program control transfers to the interrupt address when the label has been read.

If the tape is not at load point, the job is abandoned. The label occupies ten words of core storage and may be interrogated by the programmer.

The WRLABEL Request

FORM:

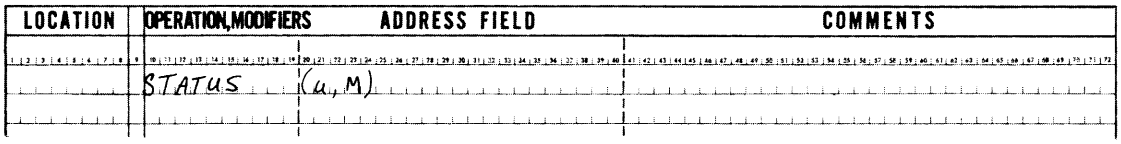
LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	WRLABEL	(u, la, ra, ia)	

The WRLABEL request writes a label onto logical unit u. The label of ten words is transmitted from the label area to the unit. If the unit is unavailable, program control transfers to the reject address. If specified, program control transfers to the interrupt address when the label has been written.

If the tape is not at load point, the job is abandoned.

The STATUS Request

FORM:



The STATUS request of a logical unit u may be taken at any time. If units are equivalenced, an M specification will request status of the master unit.

The STATUS reply is entered in the A and Q register as follows:

A register (Control Word)

Bits 47-45	Op Code
Bit 44	Jump Control
Bits 43-39	Unused
Bits 38-24	Word Count
Bits 23-18	Unused
Bits 17-0	Starting Address

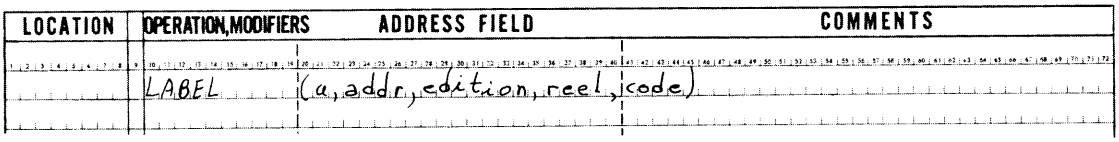
Q Register

Bit 47	Physical Unit Availability Indicator
Bit 46	Physical Unit Busy Indicator
Bit 45	Magnetic Tape Indicator
Bit 44	Bypass Indicator
Bits 43-32	Status Reply Bits
Bits 31-25	Logical Unit Number Assigned To This Physical Unit

Bit 24	Driver Indicator
Bits 23-18	Hardware Type Indicator
Bits 17-0	Control Word Address

The LABEL Request

FORM:

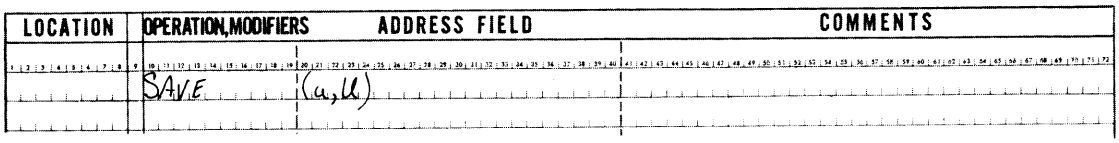


The LABEL request provides identifying information for tape labels. The parameter definitions are as follows:

u	logical unit number
addr	address of the first of two computer words containing the name.
edition	1-99. If not specified, blanks are written on the output label.
reel	1-99. If not specified, reel 1 is written on the output label.
code	000-999. For an output tape, this retention code is the number of days tape is to be saved. 999 indicates infinite retention.

The SAVE Request

FORM:

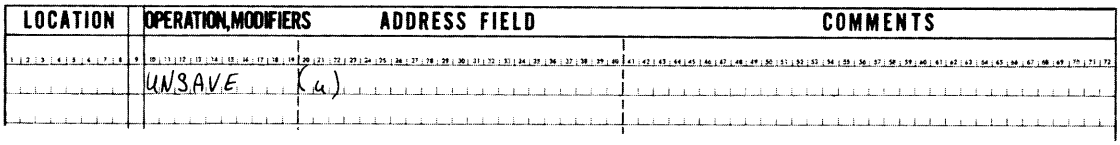


The SAVE request allows the programmer to save a tape at the completion of a job. This request may be given at any point in the program since it does not inhibit reading or writing on the unit. At the end of a job the current reel of the saved logical unit is unloaded and a message directs the operator to reserve the tape for the programmer.

U changes this request to UNSAVE.

The UNSAVE Request

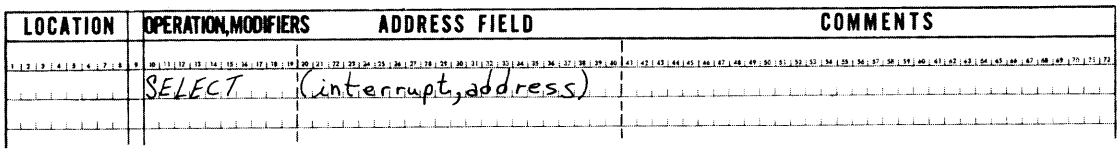
FORM:



The UNSAVE request cancels any previous SAVE request and may be given at any point in the program. The tape is released at the end of the job for subsequent use.

The SELECT Request

FORM:



The SELECT request allows the programmer to set up the interrupt system so that interrupt of the program will take place if the interrupt condition arises. The type of interrupt selected is given by the first parameter. The second parameter represents the interrupt address; i.e., the address to which program control transfers when the interrupt takes place.

The programmer selectable interrupts are the following:

Parameter	Meaning
SHIFT	Shift Fault
DIVIDE	Divide Fault
EXOV	Exponent Overflow Fault
EXUN	Exponent Underflow Fault
OVER	Fixed Point Overflow Fault
ADDR	Storage Address Fault
M1604	1604 Mode Alert
TRACE	Trace Mode Alert
INST	Illegal Instruction Fault
OPER	Operand Parity Fault
MANUAL	Manual Interrupt Alert
ABNORM	Abnormal Termination

The REMOVE Request

FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	REMOVE	(interrupt)	

The REMOVE request removes the specified interrupt. If the interrupt was not selected, REMOVE acts as a NOP.

The ABNORM Request

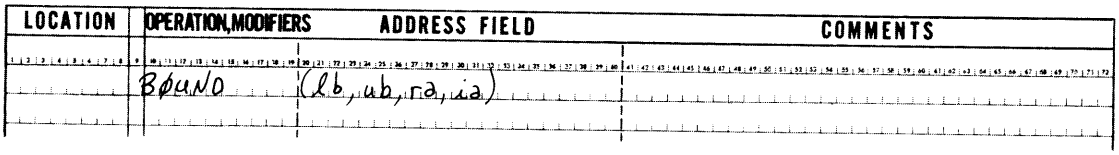
FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	ABNORM	(interrupt address)	

The ABNORM request allows the programmer to specify an alternate interrupt subroutine to be entered before Scope terminates a job for an abnormal condition. If the programmer does not specify ABNORM, Scope terminates an abnormal condition immediately.

The BOUND Request

FORM:

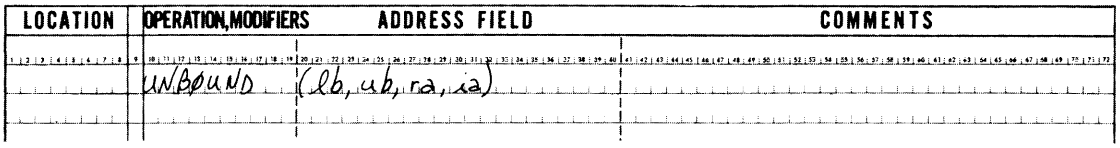


The BOUND request sets bounds outside of which any instruction reference will cause an interrupt. The first BOUND request executed in a program may set any bound range allowed by Scope. Subsequent BOUND requests must set bounds which lie within the previously set bounds. If any requested bounds overlay those previously set, the request is rejected and control is transferred to the reject address. If the bounds are accepted, the previous bounds are stored. The number of nested bounds is 5. The BOUND request has the following parameters:

lower bound upper bound	are upper and lower bounds within which the program is to operate. Bank terms may be specified. If they are not, ($) is assumed.
reject address	location to which control is transferred if the bounds list is full (5 requests) or if the requested bounds do not fall within those previously set.
interrupt address	location of the programmer's interrupt subroutine to which control is transferred when an interrupt condition occurs.

The UNBOUND Request

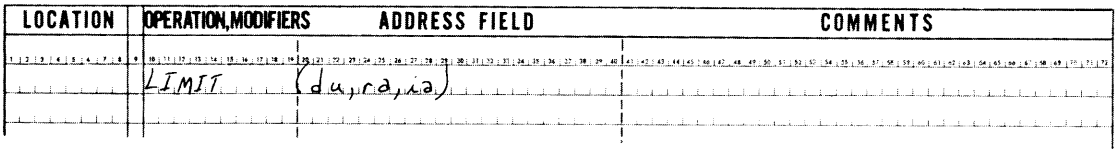
FORM:



The UNBOUND request removes the bounds set up by the last executed BOUND request and re-establishes the bounds previously set. UNBOUND cannot be used to remove the initial bounds set by Scope.

The LIMIT Request

FORM:

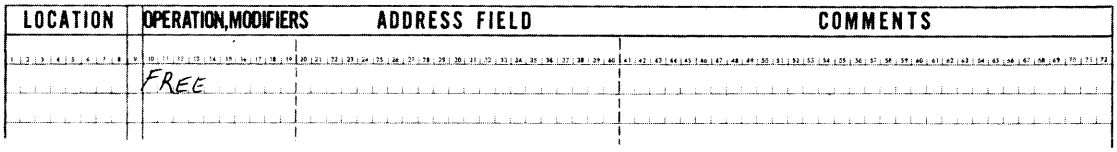


The LIMIT request sets a time limit after which control will be transferred to the interrupt address. The parameters represent the following:

du	the duration in seconds of the time limit. Milliseconds may be appended by giving the parenthesized expression (seconds, Milliseconds).
ra	the reject address if the limit is not accepted.
ia	the location to which control is transferred when the limit is reached. The interrupt subroutine is entered by a bank return jump. The interrupt subroutine should return to the interrupt address. The reject address and interrupt address may be modified by the contents of an index register.

The FREE Request

FORM:



The FREE request releases the last time set by a LIMIT request (the smallest in the list of time limits) and re-establishes the next previous time set (the next smallest limit in the list). Limits set by Scope cannot be freed.

The TIME Request

FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	TIME		

The TIME request allows the programmer to receive the time of day in the Q register in BCD. The time of day is based upon a 24-hour clock and is given in hours (hh), minutes (mm), and seconds (ss). 12:30 P.M. would be given as 123000 in BCD.

The time is entered in Q in the format:

(Q) =

47	41	29	17	5	0
0	hh	mm	ss	0	

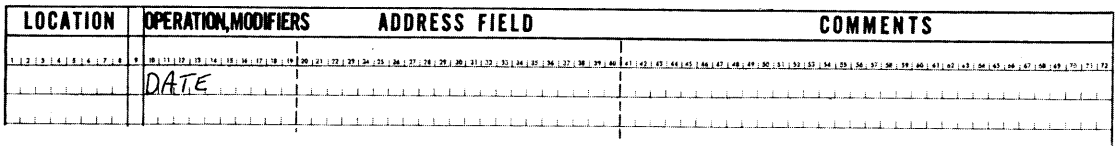
This request also returns the time remaining before the next time interrupt in the A register in binary. The time to the next interrupt is entered in A in the format:

(A) =

47	38	23	15	0
0	milliseconds	0	seconds	

The DATE Request

FORM:



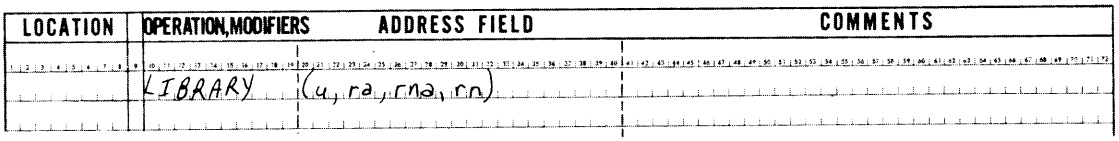
The DATE request allows the programmer to receive the month, day and year in the A register in BCD. The format is as follows:

(A) =

47	41	35	29	23	17	11	5	0
m	m	/	d	d	/	y	y	

The LIBRARY Request

FORM:



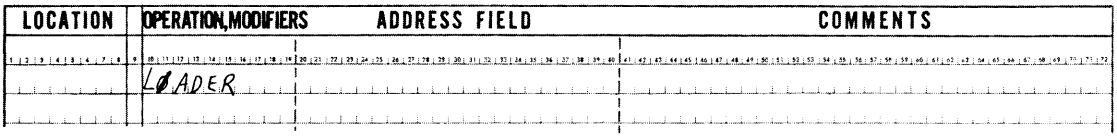
The LIBRARY request may be used to position the library tape. The programmer specifies the following parameters:

u	library logical unit number; 70 to 79 or 0.
ra	location to which control is transferred if the specified record is not found.
record name address	storage location of the first of four computer words containing the record name.
record number	signed integer, 0 through 2^{24}

Upon return to the program, the A register contains the logical unit number of the current library.

The LOADER Request

FORM:



The LOADER request is used by a running program to call the loader. If the loader is not in storage, Scope will read it from the library into its normal position following resident.

Before this request can be given properly, the A and Q registers must contain meaningful parameters. A definition of these parameters is given in the 3600 Scope Reference Manual (Pub. No. 60053300), pages 5-7.

The MEMORY Request

FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	MEMORY	(bd, ll, ul)	

The MEMORY request returns or changes the limits of available storage. The parameters are the following:

bank designator	0-7, *, or $ symbol. * designates the bank containing this request. $ symbol designates the bank in which symbol is located.
lower limit upper limit	any legal Compass address expression in the range 1 - 77776_8.

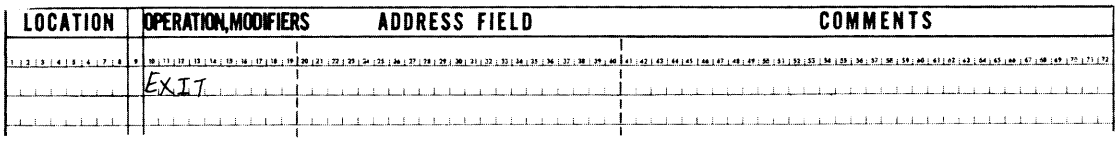
If either limit in the programmer request is zero, the other limit is not changed. If no limits are supplied in the request or if both limits are zero, the current storage limits for the specified bank will be entered in the A register in the following binary format:

(A) =

47	38	23	14	0
/	upper limit	/	lower limit	

The EXIT Request

FORM:



The EXIT request returns control from a running program to Scope. This causes a normal termination of the program as long as there are no abnormal conditions.

The HERESAQ Request

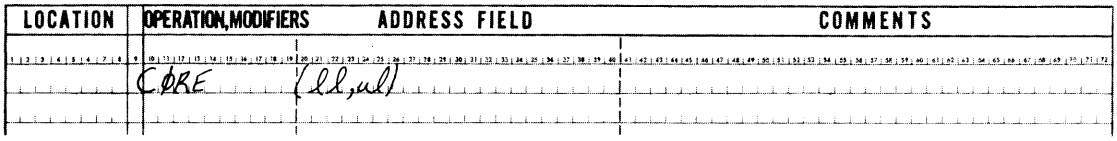
FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	HERESAQ		

The HERESAQ request allows the programmer to modify the A and Q registers in his interrupt subroutine so that when control is transferred back to the main program, Scope will enter A and Q with these values.

The CORE Request

FORM:



The CORE request allows the programmer to obtain or change the limits of available core within the bank of the subprogram. The parameters are:

lower limit upper limit	upper and lower limits within the bank.

The system macros may be used in thousands of different ways. To illustrate how some of them might be used and to show how they would be incorporated into a program consider the following problem:

Suppose a set of data cards in BCD format are to be analyzed. Each card represents information about each employee of a company. A card is punched so that;

1. Columns 1-30 give the employee's name.
2. Columns 31-40 give the area he works in.
3. Columns 41-48 give his annual salary.
4. Columns 49-64 give his social security number.
5. Columns 65-72 give his marital status.
6. Columns 73-80 give his dependents' status.

All information in a column is left justified with blank fill.

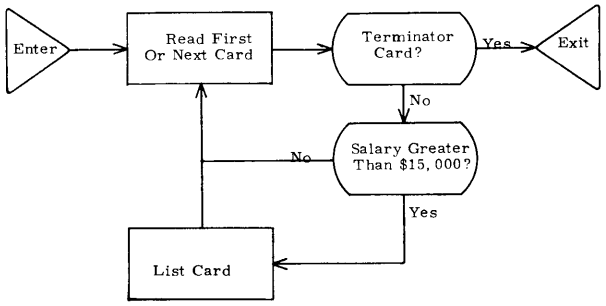
Assume:

1. The data cards are on standard input (60) between the RUN and end-of-file card.

2. The data cards are terminated with a BCD 55 punched in Columns 1 and 2.

Write a program that will list the information of each employee whose annual salary is at or above $15,000 on standard output (61).

Flowchart:



This problem could be coded in the following manner:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	IDENT	COMPARE	
	ENTRY	COMPARE	
OUTPUT	BCD	1,0	TO DOUBLE SPACE AND INDENT LIST
INPUT	BSS	10	
CWA1	IOTW	INPUT,10	
CWA2	IPTW	OUTPUT,11	TO INDENT LIST
CON1	BCD	1,15000	
COMPARE	BSS	1	
	MODE	(60,*,BCD)	SET MODE TO BCD
NEXT	READ	(60,CWA1,*)	READ ONE CARD
NOTFIN	STATUS	(60)	
	QJP,MI	NOTFIN	UNIT STILL IN OPERATION
INPFIN	LDQ	INPUT	
	ENA	0	
	LLS	12	
	INA	-505B	BCDSS
	AJP,ZR	FINISHED	DATA CARD TERMINATOR
	ENI	30,1	USED TO ALLIGN SAL. WITH 15,000
	LDQ	INPUT+5	
NEXTCHAR	ENA	0	
	LLS	6	
	RGJP,EQ	A,60B,ALLIGN	A=BLANK
	INI	-6,1	DECREASE SHIFT COUNT
	UJP	NEXTCHAR	
ALLIGN	LDA	INPUT+5	
	ARS	CON1	$15,000
	AJP,MI	NEXT	BELOW $15,000
	WRITE	(61,CWA2,*)	AT OR ABOVE $15,000
WAIT	STATUS	(61)	
	QJP,MI	WAIT	WAIT
	UJP	NEXT	
FINISHED	EXIT		
	END	COMPARE	

INTRODUCTION TO COSY

The symbol COSY represents COmpressed SYmbolic and offers another useful aid to the programmer. The compressed symbolic coding is actually a compression of the source cards with the blanks squeezed. These BCD source characters when output, are output in binary mode so that, on cards, two characters per column are punched. This reduces a deck size by a maximum of 19:1 and also reduces assembly time the next time the deck is run.

When we refer to a COSY deck, we mean the compressed symbolic source coding of one subprogram ranging from IDENT through END. A COSY deck may easily be modified using Compass instructions especially designed for COSY decks. They are; DELETE, INSERT and REPLACE. An up-to-date COSY deck can be maintained with each subsequent assembly.

COSY decks may be maintained contiguously on tape. It becomes relatively easy with the BYPASS and COSY instructions to read in and assemble any deck and to modify the deck with the INSERT, DELETE and REPLACE instructions.

GENERATING A COSY DECK

Suppose a programmer writes a long subprogram. How can he generate a COSY deck from it? A COSY deck can be generated by specifying the C option on the Compass control card. The C option means that COSY output is requested. A COSY deck can be output to magnetic tape or directly to a card punch if one is available in the system. If both the L and C options are specified, the output listing will have sequence numbers assigned to each instruction. The IDENT instruction is assigned number 1 and END is assigned number n, the nth instruction of the subprogram. The sequencing is automatic, even if no sequence numbers appear on the source cards. From these numbers the programmer can easily modify his subprogram, as will be pointed out shortly.

If C is specified, the output will be to logical unit 62 which is the standard COSY output unit. If C = xx is specified, the output will be to logical unit xx (1 to 49). Here is an example of a do-nothing program that will generate a COSY deck on tape.

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
7JØB,123,INCR,3			
3COMPASS,	L,C=20		
	IDENT	EXAMPLE	
	ENTRY	EXAMPLE	
X	BSS	1	
EXAMPLE	BSS	1	
	LDA	X	
	INA	5	
	STA	X	
	UJP	EXAMPLE	
	END		
	SCØPE		
7 7 8 8			

COSY output will be formed on logical unit 20. The standard output will look like this:

```
  BEGIN JOB AT   1018 - 35
JOB,123,INCR,3

SCOPE VERSION   6.28*

COMPASS,L,C=20
```

```
(5.1)   EXAMPLE                                    03/20/67        ED     1        PAGE NO.      1

                                          IDENT     EXAMPLE          COSY NUMBERING                  00001
       PROGRAM LENGTH              00004
       ENTRY POINTS    EXAMPLE     00001
                                          ENTRY     EXAMPLE                                          00002
         00000                  X         BSS       1                                               00003
         00001                  EXAMPLE   BSS       1                                               00004
         00002    12  0 P00000           LDA       A                                               00005
                  11  0  00005           INA       5                                               00006
         00003    20  0 P00000           STA       A                                               00007
                  75  0 P00001           UJP       EXAMPLE                                         00008
                                          END                                                       00009
          00002 SYMBOLS
```

INPUTING A COSY DECK

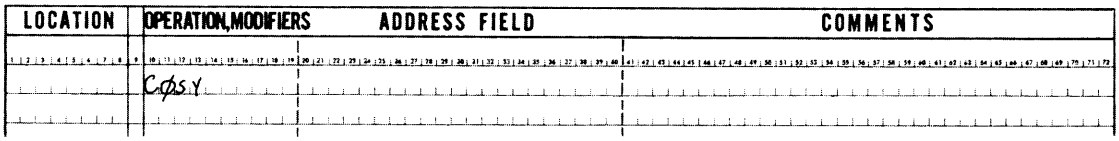
A COSY deck can be input and assembled by the programmer. In order to do so, he must specify the Y option on the Compass control card (unless the COSY deck is on standard input). If the COSY deck is on magnetic tape, the programmer must specify Y = xx, where xx represents the logical unit number (1-49). If the COSY deck is on the standard input unit, the programmer may specify Y or nothing, since standard input is assumed for all COSY input.

We now know how to define the COSY input unit, but how does the programmer read in the COSY deck, modify it, assemble it and execute it? Before we can illustrate how, we first must understand the five instructions associated with COSY. They are;

1. COSY
2. BYPASS
3. INSERT
4. DELETE
5. REPLACE

The COSY Instruction

FORM:



The COSY instruction is associated with COSY decks in that this instruction will read one COSY deck into core, translate it, perform any modification previously specified, re-number the sequence, form new COSY output (if requested), assemble it, and list the modified subprogram on the standard output unit with the new sequence numbers (if requested).

It is important to note that the COSY instruction will do this to only one COSY deck (equivalent to one subprogram - IDENT through END). If more COSY decks are to be read and assembled, each one requires another COSY instruction.

The BYPASS Instruction

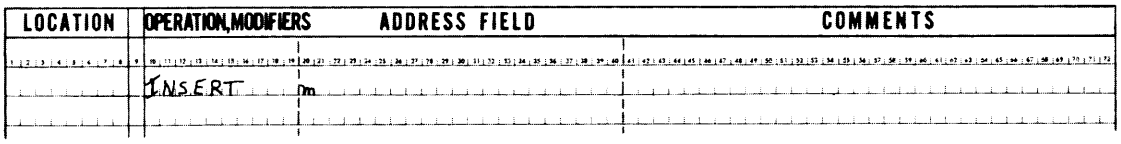
FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	BYPASS	n	

The BYPASS instruction is associated with COSY decks that are contiguously on tape. BYPASS provides a means of skipping n decks to arrive at a particular deck for processing. If BCD or COSY output is specified on the Compass control card, new Hollerith or COSY decks will be produced from the bypassed decks. After bypassing n decks, the programmer can read the next deck with a COSY instruction.

The INSERT Instruction

FORM:

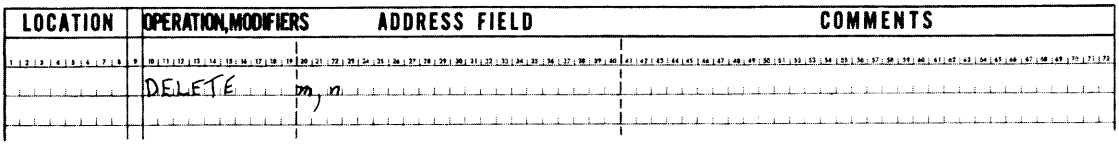


The INSERT instruction is used to add lines of coding in an existent COSY deck. INSERT causes the instructions which follow it to be inserted <u>after</u> line <u>m</u> as long as they are not COSY, BYPASS, INSERT, DELETE or REPLACE instructions.

When the first COSY instruction following this instruction is encountered, a COSY deck is read and modified accordingly. In other words the INSERT instruction would <u>precede</u> the COSY instruction for the deck that it modifies.

The DELETE Instruction

FORM:



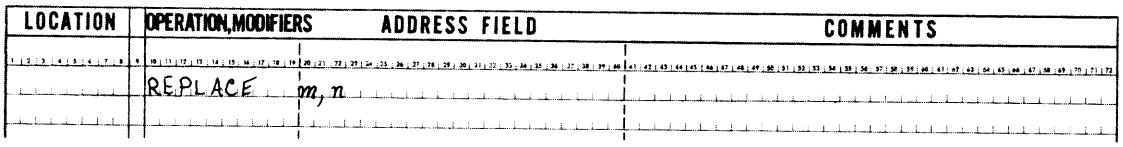
The DELETE instruction is used to modify lines of coding in an existent COSY deck. DELETE causes the deletion of coding lines m through n. A single line is deleted if only m is specified.

This instruction also allows the instructions following to be inserted in the deleted area as long as they are not COSY, BYPASS, INSERT, DELETE, or REPLACE instructions. The instructions following need not be less than, equal to, or greater than the number of instructions deleted.

When the first COSY instruction following this instruction is encountered, a COSY deck is read and modified accordingly. In other words the DELETE instructions would precede the COSY instruction for the deck that they modify.

The REPLACE Instruction

FORM:

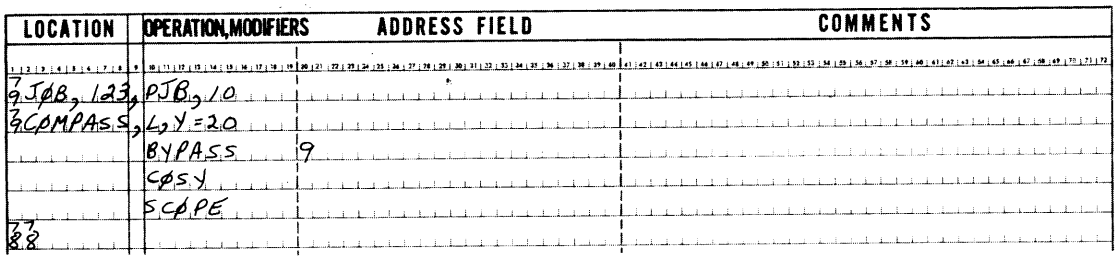


The REPLACE instruction is used to replace lines of coding in an existent COSY deck. The REPLACE instruction will do the same as the DELETE instruction.

With the previous COSY instructions in mind let us form a couple of problems in order to apply their functions. Consider the following problem:

Suppose a COSY tape contains fifty contiguous COSY decks, each deck representing a Compass subprogram. Write a program that will input the tenth deck and list it on standard output. Reference the COSY tape as logical unit 20.

This problem could be solved by coding in the following manner:



LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
7 JØB, 123,	PJB, 10		
8 COMPASS,	L, Y=20		
	BYPASS	9	
	CØSY		
	SCØPE		
77 88			

The COSY instruction reads the tenth deck, assembles it, and lists the coding on standard output.

Consider another problem:

Suppose a programmer writes three long Compass subprograms, assembles it and asks for a COSY output on logical unit 30. While holding onto his COSY tape and while scrutinizing his output listing, he notices he made a few mistakes. If the three subprograms are called A, B and C, he would like to modify them as follows:

Subprogram A:	Insert after line 15,	ENA	12B
Subprogram B:	Delete lines 25 through 31		
Subprogram C:	Insert after line 10,	LDA	XMIN
	Delete lines 14 and 15 and		
	replace with the instructions;	ENA	5
		ADD	X
		STA	Z

Write a program that will correct the three subprograms and execute them the second time he runs on the computer.

The problem could be coded in the following manner:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
$JØB,123,MØDIFY,20			
$CØMPASS,4,Y=30,X			
	INSERT	15	
	ENA	12B	12 OCTAL
	CØSY		
	DELETE	25,31	
	CØSY		
	INSERT	10	
	LDA	XMIN	
	DELETE	14,15	
	ENA	5	
	ADD	X	
	STA	Z	
	CØSY		
	SCØPE		
$LØAD			
$RUN,20,500,7			
$$			

Note how the COSY correction cards precede the COSY instruction for the deck they represent. Note also how execution immediately takes place.

OVERLAY PROCESSING

Overlay processing is a feature of the Scope system that allows execution of programs that exceed available core storage. The program is divided into independent parts, stored on tape, and called in as needed at execution time. A program may be divided into a main section and any number of overlays, each of which may contain any number of segments. Main, overlay, and segment may each contain many subprograms. However, only one main, one overlay, and one segment may be in core at any given time.

Initially control is transferred to main, which resides in core storage continuously. Main in turn can call overlays when they are needed during program execution. Segments may be called by either main or overlay.

Two questions come to mind when getting started on overlay processing:

1. How are overlay tapes prepared?
2. How are overlays called and executed?

PREPARATION OF OVERLAY TAPES

In order to prepare an overlay tape the programmer must first determine which set of subprograms he wants included under the main section, which set of subprograms he wants included under each overlay, and which set of subprograms he wants included under each segment. The main section always resides in core storage. Each overlay and its associated segment must be small enough to fit in memory core when called.

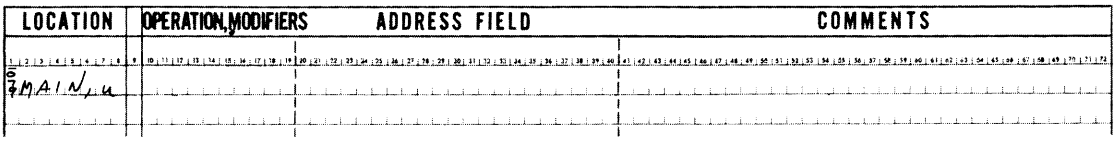
Each section is preceded by a loader control statement. A loader control statement is just like a Scope control statement (7, 9 punch in column 1) except that it contains an 11, 0, 7, 9 punch in column 1. The loader control statement defines the following set of subprograms as being a section, an overlay, or a segment. Each statement will specify on which logical unit the set of subprograms following will be written on. The subprograms may be in Compass or in relocatable binary (Compass already assembled) Each main, overlay, or segment must contain one transfer address to which program control transfers when that section is loaded.

We now define the meaning of three loader control statements.

1. MAIN
2. OVERLAY
3. SEGMENT

The MAIN Loader Control Statement

FORM:

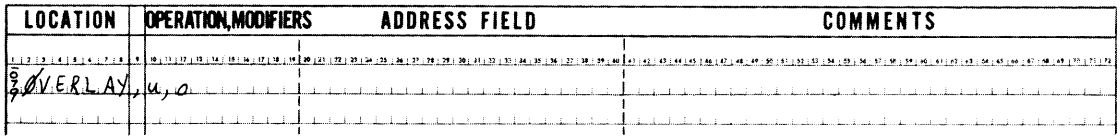


u the logical unit number of the overlay tape, 1-49, on which the main section is to be stored. u may not be omitted.

The MAIN loader control statement defines the following subprogram(s) to be a part of the main section. At load time the main section is transferred to logical unit u and to memory core. The main section represents the section to which program control transfers at the start of execution.

The OVERLAY Loader Control Statement

FORM:



u — logical unit number, 1-49, of the overlay tape on which the overlay section is to be written. u may not be omitted.

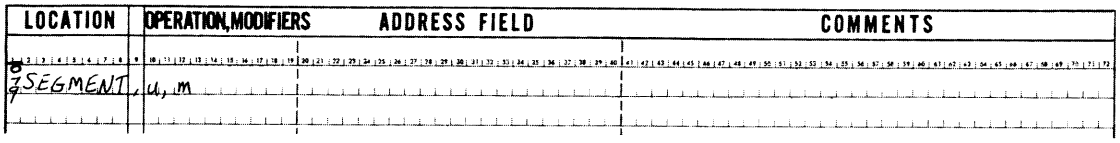
o — the decimal number (starting with 1) identifying the overlay.

The OVERLAY loader control statement defines the following subprogram(s) to be a part of an overlay section. This section is written on the tape specified by the logical unit number.

At execution time the overlay is called into memory core by a calling sequence in the main section. It is necessary for the programmer to understand this calling sequence so that he can call the overlay and transfer control to it. This is explained later.

The SEGMENT Loader Control Statement

FORM:



u logical unit number, 1-49, of the overlay tape on which the segment is to be written. u may not be omitted.

n decimal number (starting with 1) identifying the segment.

The SEGMENT loader control statement defines the following subprogram (s) to be part of a segment section. This section is written on the tape specified by the logical unit number.

At execution time the overlay is called into memory core by a calling sequence in the main or associated overlay section. The calling sequence is explained later.

In order for the Scope loader to process subprograms and form the overlay tape the sequence of input must be in the following form:

```
11
 0 MAIN
 7
 9  (Relocatable binary subprogram (s))

11
 0 OVERLAY
 7
 9 (Relocatable binary subprogram (s))

11
 0 SEGMENT
 7
 9 (Relocatable binary subprogram (s))
```

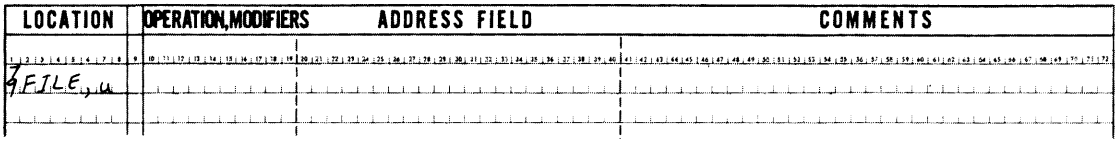
If this sequence were on standard input (60), a JOB card at the beginning, and a RUN and end-of-file card at the end of the deck would suffice. Scope would process the JOB card, the Scope loader would process, write the overlay tape and load main into memory core, and Scope would run the program by transferring control to main's transfer address. Main in turn can call overlay and either main or overlay can call segment when it is needed.

Many times, however, the programmer's subprograms are not in relocatable binary form. So the next question is, "How is the source deck formed into relocatable binary if the programmer wants to run a Compass program?" In order to show this, we must know the definition of two new statements.

1. FILE
2. FILE END

The FILE Statement

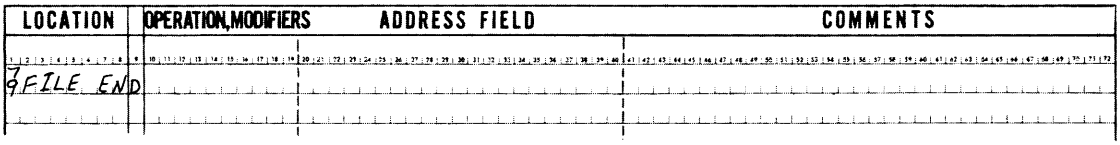
FORM:



The FILE statement is a Scope statement that allows all records following up to the FILE END statement to be transferred to logical unit u, where u may be 1-59 or 69 (load-and-go). The records are written in odd parity (binary).

The FILE END Statement

FORM:



The FILE END statement signifies the end of a FILE-FILE END sequence that transfers the binary data. Any binary data may be transferred except the Scope Control statements; SEQUENCE, JOB, END REEL, and END SCOPE.

The reason it is necessary to know these two statements is that the MAIN, OVERLAY, and SEGMENT statements must be transferred to the load-and-go unit just previous to the binary subprograms that they represent. The binary subprograms then follow each statement as a result of the Compass assemblies. Once the load-and-go tape has been formed, the LOAD statement will cause the loader to load the main section and then write the main section and all other sections on the overlay tape. The RUN statement will then execute the program.

Consider the following example showing the sequence needed to have a Compass program assembled and run with separate parts:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	1 2 3 4 5 6 7 8	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40	41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
7	9JØB,123,	DJESS,30	
7	9FILE,69		
6	3MAIN,10		
7	9FILEEND		
7	9CØMPASS,	L,X	
		.	
		(Compass subprogram(s))	
		.	
		SCØPE	
7	9FILE,69		
6	3ØVERLAY,	10,1	
7	9FILEEND		
7	9CØMPASS,	L,X	
		.	
		(Compass subprogram(s))	
		.	
		SCØPE	
7	9FILE,69		
6	3SEGMENT,	10,1	
7	9FILEEND		
7	9CØMPASS,	L,X	
		.	
		(Compass subprogram(s))	
		.	
		SCØPE	
7	9LØAD		
2	9RUN,20,1	000,4	
		(Data cards)	
7	88		

In this example the sequence of events is as follows:

1. The JOB card is processed by Scope.
2. All card images between the FILE-FILE End control statements are transferred to logical unit 69 (load-and-go). In this case the binary loader control statement MAIN, 10 is transferred.
3. The COMPASS control card causes Scope to load the Compass assembler which assembles the Compass subprograms. As the subprograms are assembled, the relocatable binary decks (object program) are output to logical unit 69 (X option).
4. The process continues until just before encountering the LOAD control statement, logical unit 69 looks like this:

```
-
0 MAIN
7
9   (Relocatable binary subprograms)

-
0 OVERLAY, 10, 1
7
9 (Relocatable binary subprograms)

-
0 SEGMENT, 10, 1
7
9 (Relocatable binary subprograms)
```

5. The LOAD statement will cause the Scope loader to;
   a. rewind logical unit 69,
   b. load the main section into core and transfer it to logical unit 10,
   c. transfer the overlay section to logical unit 10, and
   d. transfer the segment section to logical unit 10.
6. The RUN statement begins execution at the transfer address of main.

At some time the main section may wish to call in the overlay from logical unit 10. What are the necessary instructions to do this? How do you call it in and how do you transfer control to it? This is the topic now covered.

THE CALLING AND EXECUTION OF OVERLAYS

Once the overlay tape has been prepared and execution has begun, it is relatively easy to call the overlay and execute it. The main section must include a calling sequence to a standard Scope routine called LOVER (Load OVERlay) specifying the logical unit number and the overlay or segment number. The LOVER routine only loads the overlay.

The main section declares LOVER an external symbol and does a bank return jump (BRTJ) to it. Immediately following the BRTJ the programmer must prestore the necessary parameters: the logical unit number and the overlay or segment number. The form looks like this:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
ℓ	BRTJ	(*)LØVER,,*	
	EXT	LØVER	
ℓ+1	VFD	A6/N,Ø3/0,A15/V,Ø9/0,A15/S	
ℓ+2	return		

The VFD pseudo instruction has been defined in Volume II. In this case;

N = logical unit number of overlay tape

V = overlay number

S = segment number; 0 if loading an overlay

Another way of forming the same calling sequence is to write it using the CALL pseudo instruction.

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
ℓ	CALL	LØVER	
ℓ+1	VFD	A6/N,Ø3/0,A15/V,Ø9/0,A15/S	
ℓ+2	return		

In order to load overlay 1 as was written in the previous problem, it could be done thus:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	CALL	LØVER	
	VFD	A6/10,Ø3/0,A15/L,Ø9/0,A15/0	
	return		

The LOVER routine will load the overlay. If the overlay was loaded correctly, the return will leave the A register equal to zero and a bank return jump to the transfer address in the Q register. All the programmer must do is check A for zero, and if it is, execute the bank return jump instruction. The sequence might look like this:

```
LOCATION    OPERATION,MODIFIERS    ADDRESS FIELD                    COMMENTS
            CALL                   LØVER                            LØAD ØVERLAY
            VFD                    A6/10,ø3/0,A15/1,ø9/0,A15/0
            AJP,ZR                 ERRØR
            STQ                    TRANSCØNT
TRANSCØNT   ØCT                    0                                JUMP TØ ØVERLAY
            return from overlay
```

If later in the main section a calling sequence called another overlay, the first one would be destroyed since only one overlay and one segment associated with that overlay may reside in core at one time. But this shows you that they can be called and they can be executed when needed.

More can be written on this subject than is given here, but this much information gives some insights on how to get started with overlay processing. More information can be found in the 3600 Scope Reference Manual, page 6-1 (publication no. 60053300).

LIBRARY PREPARATION AND MAINTENANCE
INTRODUCTION

PRELIB is a routine on the library tape that will prepare and maintain library tapes. Library tapes include binary routines, binary data, BCD data, and end-of-file marks. Routines may be extracted from existing library tapes or they may be taken from other logical units when preparing a new library tape.

When we think about library processing, three questions come to mind.

1. How can we find out the contents and the order of contents on a library tape?
2. How can we change a present library tape?
3. How can we prepare a new library tape?

These questions serve as the next three topics. As each topic is encountered, the new control statements needed to answer the question and solve the problem are defined.

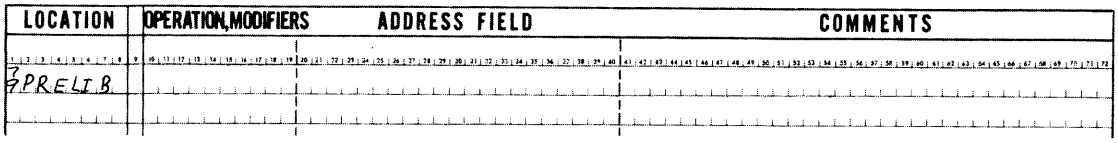
LISTING A LIBRARY TAPE

A library tape contains a set of binary records. Each record serves some purpose when monitoring, compiling, assembling, or executing a program, and each is called and used only when needed. A table of contents of the library can be listed on standard output by a short program using a library routine called PRELIB. To write the program three new control statements must be used. They are;

1. PRELIB
2. LIST
3. FINISH

The PRELIB Control Statement

FORM:



The PRELIB control statement is read by Scope. It causes Scope to load the PRELIB routine and transfer control to it. PRELIB then reads and interprets all control statements following until it reads the FINISH control statement. It then returns control to Scope for the next control statement.

The PRELIB statement must be followed by the LIST, EDIT, or PREPARE control statements.

The LIST Control Statement

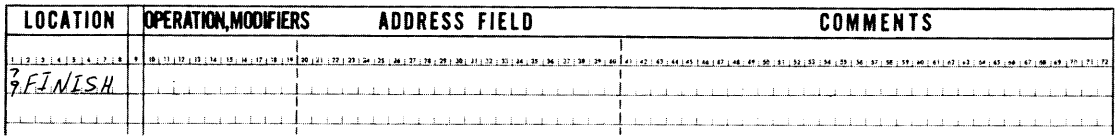
FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
7 9	LIST	(name1, name2, ...)	

The LIST control statement is read by PRELIB and is one of the three possible statements following the PRELIB statement. This statement will list the contents of the named library tape on standard output. An asterisk (*) represents the present Scope system tape.

The FINISH Control Statement

FORM:



The FINISH control statement is read by the PRELIB routine. It returns control to Scope since this signals the end of a PRELIB run.

Using these statements the table of contents could be listed on standard output using the following job:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
	SEQUENCE,1		
	JOB,1234,TRY,3.		
	PRELIB		
	LIST(*)		
	FINISH		
	8		

Below is the actual output generated from a 3600 System showing the table of contents of the library tape.

```
LABEL  5(170   TC SCOPE 6P2 80109036602

      RECORD NUMBER   NAME                    MODE

                1     BOOT                    ABS
                2     LOADER                  ABS
                3                             EOF
                4                             DIR
                5     CARPU                   REL
                6     DR3649                  REL
                                              END
                7                             EOF
                8     LOADER                  ABS
                9                             DIR
               10     PRELIB                  REL
               11     SIOPACK                 REL
               12     FTN                     REL
               13     COMPASSX                REL
               14     FTNDIAG                 BCD
               15     COMPASS                 REL
               16     MACDIR                  MDR
               17     COBOL                   REL
               18     ALDAP                   REL
               19     ALGO                    REL
               20     CDCSORT                 REL
                                              END
               21                             EOF
               22     LOADER                  ABS
               23                             DIR
               24     MACROSIM                REL
               25     MS510                   REL
               26     BUFIN                   REL
               27     GREENWCH                REL
               28     LISTNO                  REL
               29     IEVAL                   REL
               30     ICARDRD                 REL
               31     FBUFFIO                 REL
               32     IFINISH                 REL
               33     TITLESET                REL
               34     PAGE                    REL
               35     LBC                     REL
               36     IRELEASE                REL
               37     LOADSIM                 REL
               38     IOP                     REL
               39     IOS                     REL
               40     Q8QERROR                REL
               41     Q8QENTRY                REL
               42     ALLOC                   REL
               43     IOM                     REL
               44     STH                     REL
               45     TSM                     REL
               46     SLI                     REL
               47     SQRTF                   REL
               48     SINF                    REL
               49     EXPF                    REL
               50     LOGF                    REL
               51     POWRF                   REL
               52     ASINF                   REL
               53     ATANF                   REL
               54     DISKFILE                REL
               55     FORMEHR                 REL
               56     Q1QSTORE                REL
```

```
 57        Q1QREINT                        REL
 58        Q8QIFIOC                        REL
 59        XTOI                            REL
 60        RANF                            REL
 61        XFIXF                           REL
 62        ITOX                            REL
 63        TANHF                           REL
 64        ITOJ                            REL
 65        MAX1F                           REL
 66        Q2QLOADA                        REL
 67        Q8QLOADA                        REL
 68        TANF                            REL
 69        COTF                            REL
 70        Q8QMOUF                         REL
 71        Q8QXMODF                        REL
 72        REW                             REL
 73        UNLOADN                         REL
 74        EFT                             REL
 75        IOB                             REL
 76        BSP                             REL
 77        BFI                             REL
 78        BACKSKIP                        REL
 79        Q8QIFUNI                        REL
 80        LENGTHF                         REL
 81        DEC                             REL
 82        ENC                             REL
 83        Q1QDUBLE                        REL
 84        Q1QCPLEX                        REL
 85        Q7QLODLC                        REL
 86        Q8QIFDIV                        REL
 87        DSQRT                           REL
 88        DLOG                            REL
 89        DSIN                            REL
 90        DEXP                            REL
 91        DPOWER                          REL
 92        Q8QDLDA                         REL
 93        Q2QDLDA                         REL
 94        DATAN                           REL
 95        PUN                             REL
 96        Q8QSENLT                        REL
 97        Q8QIFSSW                        REL
 98        CUBERTF                         REL
 99        DCUBRT                          REL
100        Q8INOUT4                        REL
101        SLIO4                           REL
102        CLOG                            REL
103        CSIN                            REL
104        CEXP                            REL
105        CSQRT                           REL
106        CATAN                           REL
107        CABS                            REL
108        ATAN2                           REL
109        DATAN2                          REL
110        ALOG10                          REL
111        DLOG10                          REL
112        IDINT                           REL
113        DSIGN                           REL
114        DMOD                            REL
115        DMAX1                           REL
116        Q2Q07202                        REL
117        DTOI                            REL
118        Q2Q07323                        REL


119        Q2Q07313                        REL
120        Q2Q07330                        REL
121        Q9QEVAL                         REL
122        OVERSEG                         REL
123        Q2Q07331                        REL
124        Q8QRESID                        REL
125        ABSF                            REL
126        TIMEF                           REL
127        Q8QPAUSE                        REL
128        FLOATF                          REL
129        DIMF                            REL
130        XDIMF                           REL
131        INTF                            REL
132        SIGNF                           REL
133        CMPLXCVR                        REL
134        SNGL                            REL
135        CONJG                           REL
136        DABS                            REL
137        NUMBER                          REL
138        SYMBOL                          REL
139        PLOT                            REL
140        LOVER                           REL
141        BLKAPROC                        REL
142        RDPROC                          REL
143        LEAP                            REL
144        AL36INIT                        REL
145        AL36FLT                         REL
146        AL36FIX                         REL
147        MEMFULL                         REL
148        AL36GOTO                        REL
149        AL36XTOY                        REL
150        AL36XTOI                        REL
151        AL36ITOJ                        REL
152        AL36READ                        REL
153        AL36WRIT                        REL
154        AL36OUPT                        REL
155        AL36EXVL                        REL
156        AL36LN                          REL
157        AL36ATVL                        REL
158        AL36SQVL                        REL
159        AL36SIVL                        REL
160        AL36AINM                        REL
161        AL36SQNM                        REL
162        AL36EXNM                        REL
163        AL36ENNM                        REL
164        AL36SNNM                        REL
165        AL36ACNM                        REL
166        AL36SINM                        REL
167        AL36IFIO                        REL
168        AL36IFEF                        REL
169        AL36ENFL                        REL
170        AL36BKSP                        REL
171        AL36REW                         REL
172        AL36BIRD                        REL
173        AL36INPT                        REL
174        AL36INBW                        REL
175        AL36TIME                        REL
176        VGEN                            REL
                                           END
177                                        EOF
178        LOADER                          ABS
179                                        DIR
```

180	BIDUMP	REL
181	BLKB	REL
182	OUTPUTO	REL
183	CEDIT	REL
184	ARITMOVE	REL
185	ANCHOR	REL
186	SSMERGE	REL
187	DCP	REL
188	ALGO1	REL
189	COPYX	REL
		END
190		EOF
191	LOADER	ABS
192		DIR
193	STD	REL
194	IFOBJ	REL
195	FILL1	REL
196	FILL2	REL
197	SGNTRAN	REL
198	XFER1	REL
199	XFER2A	REL
200	XFER2S	REL
201	XFER3A	REL
202	XFER3M	REL
203	XFER4	REL
204	XFER4A	REL
205	ALLIT	REL
206	SUBMOVES	REL
207	SCMUOV1	REL
208	DBINT	REL
209	BDINT1	REL
210	XFER3600	REL
211	GOIF	REL
212	ACCEPT	REL
213	DBDEPON	REL
214	SEQNCH	REL
215	OBJEDT	REL
216	GPIO	REL
217	DISPLAY	REL
218	EXAMOBJ	REL
219	ADDOVFLO	REL
220	ADJOVFLO	REL
221	RNDOVFLO	REL
222	SUBROUT	REL
223	STOP	REL
224	CURNDATE	REL
225	DUMMY1	REL
226	DUMMY2	REL
227	DUMMY3	REL
228	DUMMY4	REL
229	DUMMY5	REL
230	DUMMY6	REL
231	DUMMY7	REL
232	DUMMY8	REL
233	DUMMY9	REL
234	DUMMY10	REL
235	DUMMY11	REL
		END
236		EOF
237		EOF
238	TAPEND	EOF

From the table you can see the names of routines that may be familiar. Record 10 is the PRELIB routine. Record 12 is the FORTRAN compiler. All assemblers and compilers are generally in one area.

Object routines (routines needed during execution of a program) are also listed. Record 47 is the square root routine. Record 48 is the routine that calculates the sine of an angle. These are often used in FORTRAN.

EDITING A LIBRARY TAPE

An existing library may be edited or changed by deleting, inserting, or replacing records. If the programmer knows the record number of a routine on the library, he can replace the routine with an updated version. The updated version must be in the same mode as the routine it replaces.

In order to replace a routine the programmer must first generate the updated version on a separate unit. There are several ways of doing this. At this time we are going to describe one method, and in order to do so, we will use an example.

Suppose the routine SQRTF (record 47) needed changing. It exists on the library tape in relocatable binary form. The programmer must generate a new version of SQRTF with the proper modifications.

To start, it is necessary to obtain the Compass source coding that was used when the routine was first written. This is sometimes difficult to do. However, each data center keeps on file a COSY tape containing a copy of each library routine, but in COSY format (source code but compressed). The programmer can perform the modifications and form the new relocatable binary deck (P option on the Compass control card) in one run. You might review the COSY portion of this volume for the necessary procedure.

Once the new SQRTF routine is on a tape unit (assume logical unit 62) in relocatable binary form, it then has to be transferred to the library tape. The old record 47 must be extracted and the new one inserted. All other records on the library tape should remain as they were.

In order to replace record 47, a program must be written that will include the following new statements:

1. EDIT
2. REPLACE
3. UNIT
4. REL

The EDIT Control Statement

FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
EDIT(lib,name-ec)			

The EDIT control statement is read by PRELIB and is one of the three possible statements following the PRELIB statement. This statement is the first statement of an editing deck. The editing deck contains any number of INSERT, DELETE, and REPLACE control statements.

The parameter specifies the source library tape. If * is specified, the current Scope system library is used. The new library will have the same name as the old one with the edition number incremented by 1. All records will be copied except those that are deleted or replaced. During editing, the old directories (table of contents) are updated by PRELIB.

The FINISH control statement terminates the editing deck. When it is encountered the rest of the source library is copied. The new table of contents is written on the standard output unit.

The REPLACE Control Statement

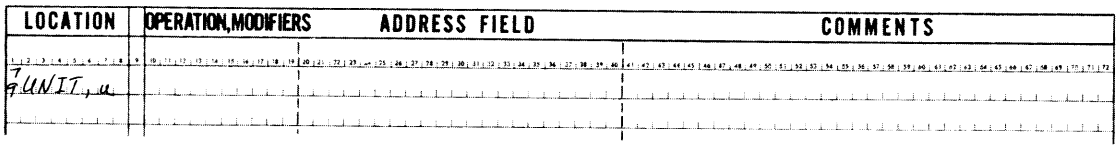
FORM:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
GREPLACE	(rec1,rec2,...)		

The REPLACE control statement will copy the source library up to the records that are to be replaced. The following control statements will determine what then will be written on the new library tape and from which unit it will come.

The UNIT Control Statement

FORM:

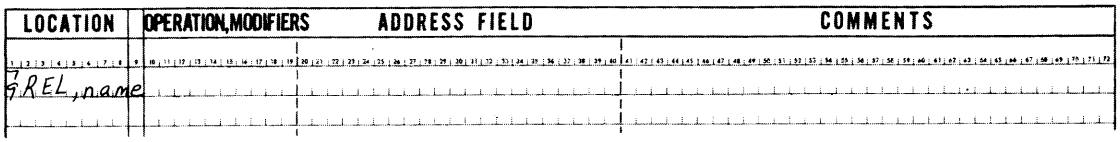


The UNIT control statement specifies from which logical unit, 1-49, 60, or 62, the input records will be read. This unit will be referenced for any subsequent input requests until another UNIT control statement changes the specification.

This statement just specifies the unit. The input record is not read until a control statement specifying mode (e. g. REL) is encountered.

The REL Control Statement

FORM:



The REL control statement is one of several control statements that specify the mode of input. REL specifies that the information on a logical unit is a single subroutine in relocatable binary card form and is to be written as one record on the new library in a condensed relocatable form.

If the programmer understands the previous control statements, he is ready to write a program that will transfer his updated SQRTF subroutine from logical unit 62 to the library tape. He can do so by the following coding:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
7SEQUENCE,1			
7JOB,1.23,NEWLIB,5			
7PRELIB			
7EDIT(*)			
7REPLACE(47)			
7UNIT,62			
7REL,SQRTF			
7FINISH			
77			

This program will;

1. Copy the source library up to record 47.
2. Transfer the relocatable binary subroutine (SQRTF) from logical unit 62.
3. Skip the old record 47 and copy the rest of the library.

PREPARING A LIBRARY TAPE

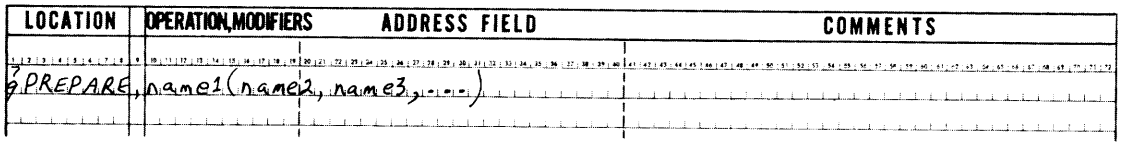
Besides editing an existing library tape, it may become necessary at times to prepare a new library tape. A new library can be formed by extracting information from an existing library and including records from other logical units.

In order to extract records from existing libraries to form a new library, two new control statements must be defined.

1. PREPARE
2. EXTRACT

The PREPARE Control Statement

FORM:

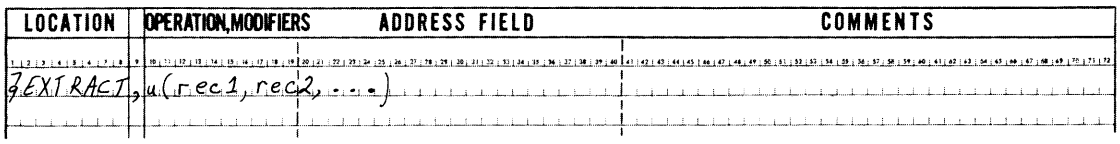


The PREPARE control statement is read by PRELIB and is one of the three possible statements following the PRELIB statement. PREPARE instructs PRELIB to prepare a new library from the source libraries.

NAME1 designates the name of the new library. If * is specified, the new library will have the same name as the current system library. NAME2, NAME3, etc., designates the names of library tapes from which information will be taken to form the new library. An * for one of these names means that the current system library can be used as a source.

The EXTRACT Control Statement

FORM:



The EXTRACT control statement specifies the records on source library tapes which are to be transferred to the new tape. The records will be transferred from logical unit u (*, 70, 72-79) onto the new tape, unit 71 (logical unit 71 is always considered the new library tape). A record may be referred to by either its name or its number as found in the table of contents.

A job to prepare a new library tape from two existing ones might look like the following:

LOCATION	OPERATION,MODIFIERS	ADDRESS FIELD	COMMENTS
7			
7 9SEQUENCE	E,1		
7 9JOB	,1234,EXAM,S		
7 9RELIB			
7 9PREPARE	,*(*,LIB1)		
7 9EXTRACT	,*(1-16)		
7 9EXTRACT	,72(CØBØL)		
7 9EXTRACT	,*(18-238)		
7 9FINISH			
7 88			

This job effectively replaces the old library with the exception that it has a new COBOL compiler.

1. A new Scope system library is prepared from the current library and LIB1. It has the same name as the current library.

2. Records 1-16 are copied from current library.

3. The COBOL compiler is copied from LIB1 (72).

4. The rest of the current library is copied.

This may give some insights on how to begin preparing or editing a library tape. Much more can be included as the reader has use for it. For more information read the "Library Preparation and Maintenance" section (P. 7-1) of the 3600 Scope Reference Manual (Pub. No. 60053300).

SECTION II

OUTPUT LISTINGS

This section contains the computer output listings generated from the example problems given in each of the three volumes. The purpose of these listings is to show how the subprograms are assembled.

The coding on the listing is taken from the coding on the coding sheet for each problem. The only difference is that a few block storage reserve (BSS) instructions may have been added at the end of each subprogram. This is to prevent a diagnostic resulting from an undefined symbol.

```
(5.1)  EVALUATE                                          03/20/67      ED    0      PAGE NO.     1

                                        IDENT    EVALUATE          VOLUME 1-SECTION III-PROBLEM 5
PROGRAM LENGTH              00006
ENTRY POINTS    EVALUATE    00000
                                        ENTRY    EVALUATE
  00000                 EVALUATE BSS     1
  00001   12  0 P00003           LDA     A
          24  0 P00004           MUI     B                 AB
  00002   14  0 P00005           ADD     C                 AB + C
          75  0 P00000           SLJ     EVALUATE
  00003                 A        BSS     1
  00004                 B        BSS     1
  00005                 C        BSS     1
                                 END
   00004 SYMBOLS
```

```
(5.1)      TEST                                          03/20/67      ED    0      PAGE NO.     1

                                        IDENT    TEST              VOLUME 1-SECTION III-PROBLEM 6
PROGRAM LENGTH              00156
ENTRY POINTS      TEST      00000
                                        ENTRY    TEST
  00000                 TEST     BSS     1
  00001   50  1  00000           ENI     0,1
          10  0  00000           ENA     0
  00002   14  1 P00012 REPEAT    ADD     SCORES,1
          50  0  00000
  00003   54  1  00143           ISK     99,1
          75  0 P00002           SLJ     REPEAT
  00004   04  0  00000           ENQ     0                 PREPARE FOR DIVIDE
          25  0 P00007           DVI     DEC100
  00005   20  0 P00010           STA     AVE
          21  0 P00011           STQ     REM
  00006   75  0 P00000           SLJ     TEST              RETURN
          50  0  00000
  00007   00  0  00000 DEC100    DEC     100
          00  0  00144
  00010                 AVE      BSS     1
  00011                 REM      BSS     1
  00012                 SCORES   BSS     100
                                 END
   00006 SYMBOLS
```

```
(5.1)  INTEREST                                          03/20/67      ED    0       PAGE NO.      1

                                          IDENT    INTEREST          VOLUME 1-SECTION III-PROBLEM 7
PROGRAM LENGTH              23445
ENTRY POINTS    INTEREST    00005
                                          ENTRY    INTEREST
  00000   00  0  00000  DOC      DEC      0                  PRESTORE ZEROS
          00  0  00000
  00001   00  0  00000  LAW      DEC      0                  AT THESE
          00  0  00000
  00002   00  0  00000  FIRE     DEC      0                  LOCATIONS
          00  0  00000
  00003   00  0  00000  GARBAGE  DEC      0
          00  0  00000
  00004   00  0  00000  SALES    DEC      0
          00  0  00000
  00005                INTEREST BSS      1
  00006   50  1  23420           ENI      10000,1
          10  0  00001           ENA      1
  00007   64  1 P00025  A        EQS      NUM,1              SEARCH FOR DOCTOR
          75  0 P00011           SLJ      D                  NO MORE DOCTORS
  00010   72  0 P00000           RAO      DOC                FOUND A DOCTOR
          75  0 P00007           SLJ      A                  CONTINUE SEARCH
  00011   50  1  23420  B        ENI      10000,1
          10  0  00002           ENA      2
  00012   64  1 P00025           EQS      NUM,1              SEARCH FOR LAWYERS
          75  0 P00014           SLJ      C                  NO MORE LAWYERS
  00013   72  0 P00001           RAO      LAW                FOUND A LAWYER
          75  0 P00011           SLJ      B                  CONTINUE SEARCH
  00014   50  1  23420  C        ENI      10000,1
          10  0  00003           ENA      3
  00015   64  1 P00025           EQS      NUM,1              SEARCH FOR FIREMEN
          75  0 P00017           SLJ      D                  NO MORE FIREMEN
  00016   72  0 P00002           RAO      FIRE               FOUND A FIREMEN
          75  0 P00014           SLJ      C                  CONTINUE SEARCH
  00017   50  1  23420  D        ENI      10000,1
          10  0  00004           ENA      4
  00020   64  1 P00025           EQS      NUM,1              SEARCH FOR GARBAGEMEN
          75  0 P00022           SLJ      E                  NO MORE GARBAGEMEN
  00021   72  0 P00003           RAO      GARBAGE            FOUND A GARBAGEMAN
          75  0 P00017           SLJ      D                  CONTINUE SEARCH
  00022   50  1  23420  E        ENI      10000,1
          10  0  00005           ENA      5
  00023   64  1 P00025           EQS      NUM,1              SEARCH FOR SALESMEN
          75  0 P00005           SLJ      INTEREST           NO MORE SALESMEN
  00024   72  0 P00004           RAO      SALES              FOUND A SALESMAN
          75  0 P00022           SLJ      E                  CONTINUE SEARCH
  00025                NUM      BSS      10000
                                END

   00014 SYMBOLS
```

```
(5.1)  ENTRANT                                           03/20/67      ED    0       PAGE NO.      1

                                          IDENT    ENTRANT           VOLUME 1-SECTION III-PROBLEM 8
PROGRAM LENGTH              00012
ENTRY POINTS    ENTRANT     00002
                                          ENTRY    ENTRANT
  00000                C        BSS      1
  00001   11  1  11111  CHECK    OCT      1111111111111111
          11  1  11111
  00002                ENTRANT  BSS      1
  00003   37  0 P00001           SSH      CHECK              1ST, 2ND, OR 3RD
          75  0 P00006           SLJ      APLB               1ST OR 2ND
  00004   12  0 P00010           LDA      A                  3RD
          15  0 P00011           SUB      B
  00005   20  0 P00000           STA      C                  C = A - B
          75  0 P00002           SLJ      ENTRANT
  00006   12  0 P00010  APLB     LDA      A
          14  0 P00011           ADD      B
  00007   20  0 P00000           STA      C                  C = A + B
          75  0 P00002           SLJ      ENTRANT
  00010                A        BSS      1
  00011                B        BSS      1
                                END

   00006 SYMBOLS
```

```
(5.1)  ENTREXAM                                          03/20/67      ED    0       PAGE NO.     1

                                          IDENT     ENTREXAM          VOLUME 1-SECTION III-PROBLEM 9
PROGRAM LENGTH              11615
ENTRY POINTS    ENTREXAM    00001
                                          ENTRY     ENTREXAM
  00000                     HANK      HSS     1
  00001                     ENTREXAM  HSS     1
  00002   04  0  00777               ENQ     777B              OCTAL 777
          06  0  00030               QLS     24
  00003   44  1 P00005               LDL     INFORM,1
          01  0  00030               ARS     24
  00004   20  0 P00000               STA     HANK
          75  0 P00001               SLJ     ENTREXAM
  00005                     INFORM    HSS     5000
                                    END
   00003 SYMBOLS
```

```
(5.1)  SEQUENCE                                          03/20/67      ED    0       PAGE NO.     1

                                          IDENT     SEQUENCE          VOLUME 1-SECTION III-PROBLEM 10
PROGRAM LENGTH              00013
ENTRY POINTS    SEQUENCE    00001
                                          ENTRY     SEQUENCE
  00000                     SEQ1      HSS     1
  00001                     SEQUENCE  HSS     1
  00002   50  1  00000               ENI     0,1
          12  0 P00017               LDA     SEQ+9             10TH TERM
  00003   26  1 P00020   REPEAT      MUF     SEQ+10,1
          50  0  00000
  00004   54  1  00003               ISK     3,1
          75  0 P00003               SLJ     REPEAT
  00005   20  0 P00000               STA     SEQ1
          75  0 P00001               SLJ     SEQUENCE
  00006                     SEQ       HSS     5
                                    END
   00004 SYMBOLS
```

```
(5.1)    MATRIX                                          03/20/67        ED    0        PAGE NO.       1

                                          IDENT     MATRIX              VOLUME 1-SECTION III-PROBLEM 11
      PROGRAM LENGTH              01757
      ENTRY POINTS    MATRIX      00002
                                          ENTRY     MATRIX
        00000                     RANGE   BSS       1
        00001                     DIVIS   BSS       1
        00002                     MATRIX  BSS       1
        00003   12  0 P00007              LDA       TERM + 61           = 62ND TERM
                26  0 P00007              MUF       TERM + 69           = 70TH TERM
        00004   20  0 P00001              STA       DIVIS
                12  0 P00007              LDA       TERM + 59           = 60TH TERM
        00005   26  0 P00007              MUF       TERM + 71           = 72ND TERM
                27  0 P00001              DVF       DIVIS
        00006   20  0 P00000              STA       RANGE
                75  0 P00002              SLJ       MATRIX              RETURN
        00007                     TERM    BSS       1000
                                          END
         00004 SYMBOLS
```

```
(5.1)    EVAL                                            03/20/67        ED    0        PAGE NO.       1

                                          IDENT     EVAL                VOLUME 1-SECTION III-PROBLEM 12
      PROGRAM LENGTH              00013
      ENTRY POINTS    EVAL        00004
                                          ENTRY     EVAL
        00000                     Z       BSS       1
        00001                     SAVE    BSS       1
        00002   20  0  54500     CON1    DEC       18.5
                00  0  00000
        00003   20  0  26314     CON2    DEC       3.2
                63  1  46315
        00004                     EVAL    BSS       1
        00005   12  0 P00002              LDA       CON1
                32  0 P00011              FMU       X                   18.5X
        00006   20  0 P00001              STA       SAVE
                12  0 P00003              LDA       CON2
        00007   32  0 P00012              FMU       Y                   3.2Y
                30  0 P00001              FAD       SAVE
        00010   20  0 P00000              STA       Z
                75  0 P00004              SLJ       EVAL
        00011                     X       BSS       1
        00012                     Y       BSS       1
                                          END
         00007 SYMBOLS
```

```
(5.1)  FORMULA                                          03/20/67      ED    0       PAGE NO.     1

                                                IDENT     FORMULA       VOLUME I-SECTION III-PROBLEM 13
PROGRAM LENGTH                    00020
ENTRY POINTS     FORMULA          00005
                                                ENTRY     FORMULA
  00000                           SAV1      BSS     1
  00001                           SAV2      BSS     1
  00002   20  1  35670            CON1      DEC     1500.
          00  0  00000
  00003   17  7  36314            CON2      DEC     .05
          63  1  46315
  00004                           W         BSS     1
  00005                           FORMULA   BSS     1
  00006   12  0 P00015                     LDA     R
          32  0 P00017                     FMU     T
  00007   30  0 P00016                     FAD     S
          20  0 P00000                     STA     SAV1          = S + RT
  00010   12  0 P00017                     LDA     T
          32  0 P00002                     FMU     CON1
  00011   20  0 P00001                     STA     SAV2          = 1500T
          12  0 P00015                     LDA     R
  00012   31  0 P00001                     FSB     SAV2
          32  0 P00003                     FMU     CON2          =.05(R-1500T)
  00013   33  0 P00000                     FDV     SAV1
          20  0 P00004                     STA     W             ANSWER
  00014   75  0 P00005                     SLJ     FORMULA
          50  0  00000
  00015                           R         BSS     1
  00016                           S         BSS     1
  00017                           T         BSS     1
                                            END

   00011 SYMBOLS
```

```
(5.1)  FUNCTION                                         03/20/67      ED    0       PAGE NO.     1

                                                IDENT     FUNCTION      VOLUME I-SECTION III-PROBLEM 14
PROGRAM LENGTH                    00015
ENTRY POINTS     FUNCTION         00003
                                                ENTRY     FUNCTION
  00000   20  0  26000  CON1      DEC     3.
          00  0  00000
  00001   20  0  14000  CON2      DEC     1.
          00  0  00000
  00002                  SAVE      BSS     1
  00003                  FUNCTION  BSS     1
  00004   12  0 P00014             LDA     A1
          32  0 P00000             FMU     CON1
  00005   20  0 P00002             STA     SAVE          = 3*X1
          12  0 P00014             LDA     A1
  00006   32  0 P00014             FMU     A1
          30  0 P00002             FAD     SAVE
  00007   31  0 P00001             FSB     CON2
          20  0 P00002             STA     SAVE          = (X1)**2*3*X1-1
  00010   77  1  00004             LDA,MG  SAVE
          12  0 P00002
  00011   31  0 P00001             FSB     CON2
          22  2 P00013             AJP,PL  GREATER
  00012   10  0  00000  LESS       ENA     0
          75  0 P00003             SLJ     FUNCTION
  00013   10  0  00077  GREATER    ENA     77B           77 OCTAL
          75  0 P00003             SLJ     FUNCTION
  00014                  X1        BSS     1
                                    END

   00007 SYMBOLS
```

```
(5.1)  MATRIXMU                                          03/20/67      ED    0       PAGE NO.      1

                                          IDENT     MATRIXMU          VOLUME I-SECTION III-PROBLEM 15

       PROGRAM LENGTH              00127
       ENTRY POINTS    MATRIXMU    00000
                                          ENTRY     MATRIXMU
         00000                 MATRIXMU   BSS       1
         00001    50  1  00000           ENI       0,1               1ST EL. OF FIRST COL.
                  50  2  00110           ENI       72,2              1ST EL. OF LAST COL.
         00002    12  1 P00006  A        LDA       ARRAY,1
                  50  0  00000
         00003    77  1  20040           FAD,RP    ARRAY,1,2 \       (AX1)+(AX9)  AX9
                  30  1 P00006
         00004    54  1  00010           ISK       8,1               9 ELEMENTS
                  75  0 P00002           SLJ       A                 NO
         00005    75  0 P00000           SLJ       MATRIXMU          YES, RETURN
                  50  0  00000
         00006                 ARRAY      BSS       81
                                          END
          00003 SYMBOLS
```

```
(5.1)  SEPARATE                                          03/20/67      ED    0       PAGE NO.      1

                                          IDENT     SEPARATE          VOLUME I-SECTION III-PROBLEM 16

       PROGRAM LENGTH              01154
       ENTRY POINTS    SEPARATE    00764
                                          ENTRY     SEPARATE
         00000                 UNSAT      BSS       100
         00144                 POOR       BSS       100
         00310                 FAIR       BSS       100
         00454                 GOOD       BSS       100
         00620                 EXCELENT   BSS       100
         00764                 SEPARATE   BSS       1
         00765    50  1  00000           ENI       0,1
                  50  2  00000           ENI       0,2
         00766    50  3  00000           ENI       0,3
                  50  4  00000           ENI       0,4
         00767    50  5  00000           ENI       0,5
                  50  6  00000           ENI       0,6
         00770    12  6 P01010  NEXT     LDA       SCORES,6
                  50  0  00000
         00771    62  0  00106           RGJP,LT   A,70,LT70          (A) LT70
                  21  3 P01000
         00772    62  0  00114           RGJP,LT   A,76,LT76          (A) LT76
                  21  3 P01002
         00773    62  0  00133           RGJP,LT   A,91,LT91          (A) LT91
                  21  3 P01004
         00774    62  0  00140           RGJP,LT   A,96,LT96          (A) LT96
                  21  3 P01006
         00775    20  5 P00620  GE96     STA       EXCELENT,5
                  51  5  00001           INI       1,5
         00776    54  6  00143  TEST     ISK       99,6              100 SCORES TESTED
                  75  0 P00770           SLJ       NEXT              NO
         00777    75  0 P00764           SLJ       SEPARATE          YES
                  50  0  00000
         01000    20  1 P00000  LT70     STA       UNSAT,1
                  51  1  00001           INI       1,1
         01001    75  0 P00776           SLJ       TEST
                  50  0  00000
         01002    20  2 P00144  LT76     STA       POOR,2
                  51  2  00001           INI       1,2
         01003    75  0 P00776           SLJ       TEST
                  50  0  00000
         01004    20  3 P00310  LT91     STA       FAIR,3
                  51  3  00001           INI       1,3
         01005    75  0 P00776           SLJ       TEST
                  50  0  00000
         01006    20  0 P00454  LT96     STA       GOOD, 4
                  51  4  00001           INI       1,4
         01007    75  0 P00776           SLJ       TEST
                  50  0  00000
         01010                 SCORES     BSS       100
                                          END
          00016 SYMBOLS
```

```
(5.1)   UNPACK                                          03/20/67      ED    0       PAGE NO.      1

                                        IDENT     UNPACK              VOLUME 1-SECTION III-PROBLEM 17
PROGRAM LENGTH              00021
ENTRY POINTS    UNPACK      00000
                                        ENTRY     UNPACK
  00000                     UNPACK      BSS       1
  00001   50  1  00000                 ENI       0,1
          50  2  00052                 ENI       42,2                MEM. OFFSET DES.
  00002   63  1  20006     A           LBYT,Q0,E6,RI,CL  CARD,1,2
          52  2 P00007
  00003   50  2  00052                 ENI       42,2                EXECUTED ONLY WHEN
          51  1  00001                 INI       1,1                 COMPLETE WORD EXAMINED
  00004   62  0  00073                 RGJP,EQ   Q,738,UNPACK        SKIP EXIT TO HERE
          01  4 P00000
  00005   06  0  00052                 QLS       42
          07  0  00006                 LLS       6                   CHAR TO A
  00006   75  0 P00002                 SLJ       A                   NEXT CHAR
          50  0  00000
  00007                     CARD        BSS       10
                                        END
   00003 SYMBOLS
```

```
(5.1)       TAX                                         03/20/67      ED    0       PAGE NO.      1

                                        IDENT     TAX                 VOLUME 1-SECTION III-PROBLEM 18
PROGRAM LENGTH              47054
ENTRY POINTS    TAX         23420
                                        ENTRY     TAX
  00000                     TAXABLE     BSS       10000
  23420                     TAX         BSS       1
  23421   50  1  00000                 ENI       0,1
          50  2  00000                 ENI       0,2
  23422   12  1 P23434     NEXT        LDA       EMPLOY,1
          50  0  00000
  23423   63  0  26057                 NBJP      A,47,A
          60  0 P23426
  23424   54  1  23417     LAST        ISK       9999,1              LAST EMPLOYEE
          75  0 P23422                 SLJ       NEXT                NO
  23425   75  0 P23420                 SLJ       TAX                 YES
          50  0  00000
  23426   63  0  26056     A           NBJP      A,46,B
          60  0 P23430
  23427   75  0 P23424                 SLJ       LAST
          50  0  00000
  23430   63  0  26055     B           ZBJP      A,45,ENTER          FOUND ONE
          64  0 P23432
  23431   75  0 P23424                 SLJ       LAST
          50  0  00000
  23432   20  2 P00000     ENTER       STA       TAXABLE,2
          51  2  00001                 INI       1,2
  23433   75  0 P23424                 SLJ       LAST
          50  0  00000
  23434                     EMPLOY      BSS       10000
                                        END
   00010 SYMBOLS
```

```
(5.1)    INPUT                                          03/20/67        ED     0        PAGE NO.       1

                                                IDENT     INPUT              VOLUME 1-SECTION III-PROBLEM 19
      PROGRAM LENGTH                 00022
      ENTRY POINTS    INPUT          00014
                                                ENTRY     INPUT
        00000                        CARD       BSS       10
        00012   10  0  00012        CWA1       IOTW      CARD,10            8 CHAR PER WORD
                00  0 P00000
        00013   10  0  00077        REJ        ENA       77B
                50  0  00000
        00014                        INPUT      BSS       1
        00015   74  0 P00013                   CONN      0,2,3,REJ
                00  0  02003
        00016   74  1 P00013                   EXTF      0,3,REJ            556BPI
                00  0  00003
        00017   74  1 P00013                   EXTF      0,2,REJ            BCD FORMAT
                00  0  00002
        00020   74  2 P00013                   BEGR      0,CWA1,REJ
                00  0 P00012
        00021   75  0 P00014                   SLJ       INPUT
                50  0  00000
                                                END
         00004 SYMBOLS
```

```
(5.1)    ARITHFUN                                       03/20/67        ED     0        PAGE NO.       1

                                                IDENT     ARITHFUN           VOLUME 1-SECTION III-PROBLEM 20
      PROGRAM LENGTH                 00005
      ENTRY POINTS    ARITHFUN       00000
                                                ENTRY     ARITHFUN
        00000                        ARITHFUN   BSS       1
        00001   63  0  36001                   NBJP,ST   IM,1,*+1           SET DIV. FAU. BIT
                61  0 P00002
        00002   63  0  36004                   NBJP,ST   IM,4,*+1           SET ARITH. OV. BIT
                61  0 P00003
        00003   77  0  00022                   INF       18                 SET INT. ACT.
                50  0  00000                                                CONT. PROG.

        00004   75  0 P00000                   SLJ       ARITHFUN           RETURN
                50  0  00000
                                                END
         00001 SYMBOLS
```

```
(5.1)      TEST                                                         03/20/67       ED      0        PAGE NO.      1
                                             IDENT     TEST               VOLUME II-PROBLEM 1
     PROGRAM LENGTH                   01757
     ENTRY POINTS      TEST           00000
                                             ENTRY     TEST
       00000                        TEST     BSS       1
       00001     04  0   00000               ENQ       0
                 50  1   00000               ENI       0,1
       00002     12  1 P00007       NEXT     LDA       TAB,1
                 22  3 P00004                AJP,MI    MINUS              POS OR NEG
       00003     21  1 P00007       PLUS     STQ       TAB,1              POS
                 50  0   00000
       00004     77  1   00020      MINUS    STA,CM    TAB,1              NEG
                 20  1 P00007
       00005     54  1   23417               ISK       9999,1
                 75  0 P00002                SLJ       NEXT
       00006     75  0 P00000                SLJ       TEST
                 50  0   00000
       00007                        TAB      BSS       1000
                                             END
        00005 SYMBOLS
```

```
(5.1)  TRANSFER                                                         03/20/67       ED      0        PAGE NO.      1
                                             IDENT     TRANSFER           VOLUME II-PROBLEM 2
     PROGRAM LENGTH                   00010
     ENTRY POINTS      TRANSFER       00001
                                             ENTRY     TRANSFER
       00000                        TEMP     BSS       1
       00001                        TRANSFER BSS       1
       00002     60  0 P00006                SAU       TRANSMIT           INITIAL ADDRESS
                 01  0   00030               ARS       24
       00003     61  0 P00006                SAL       TRANSMIT           TERMINAL ADDRESS
                 21  0 P00000                STQ       TEMP
       00004     53  1 P00000                LIL       TEMP,1             WORD COUNT
                 50  0   00000
       00005     55  1 P00006       CHECKCNT IJP       TRANSMIT,1
                 75  0 P00001                SLJ       TRANSFER
       00006     12  1   77777      TRANSMIT LDA       **,1
                 20  1   77777               STA       **,1
       00007     75  0 P00005                SLJ       CHECKCNT
                 50  0   00000
                                             END
        00004 SYMBOLS
```

```
(5.1)     FILE                                          03/20/67      ED     0       PAGE NO.      1

                                        IDENT     FILE              VOLUME II-PROBLEM 3
PROGRAM LENGTH              01760
ENTRY POINTS    FILE        00000
                                        ENTRY     FILE
  00000                         FILE    BSS       1
  00001    50  1  00000              ENI       0,1               ADDRESS COUNTER
           50  2  00000              ENI       0,2               CHARACTER COUNTER
  00002    16  1 P00010  NEXTWORD    LDQ       BUF,1
           50  0  00000
  00003    10  0  00000  NEXTCHAR    ENA       0
           07  0  00006              LLS       6
  00004    11  0  77760              INA       -17B
           22  0 P00007              AJP,ZR    FOUND
  00005    54  2  00007              ISK       7,2               LAST CHARACTER
           75  0 P00003              SLJ       NEXTCHAR          NO
  00006    51  1  00001              INI       1,1               YES, BUMP ADDRESS COUNTER
           75  0 P00002              SLJ       NEXTWORD
  00007    12  1 P00010  FOUND      LDA       BUF,1
           75  0 P00000              SLJ       FILE
  00010                         BUF     BSS       1000
                                        END

   00005 SYMBOLS
```

```
(5.1)     SWITCH                                        03/20/67      ED     0       PAGE NO.      1

                                        IDENT     SWITCH            VOLUME II-PROBLEM 4
PROGRAM LENGTH              00152
ENTRY POINTS    SWITCH      00000
                                        ENTRY     SWITCH
  00000                         SWITCH  BSS       1
  00001    50  1  00000              ENI       0,1               COUNTER FOR FIRST WORD
           50  2  00143              ENI       99,2              COUNTER FOR LAST WORD
  00002    12  1 P00006  NEXT       LDA       TAB,1
           16  2 P00006              LDQ       TAB,2
  00003    20  2 P00006              STA       TAB,2
           21  1 P00006              STQ       TAB,1
  00004    55  2 P00005              IJP       *+1,2
           75  0 P00000              SLJ       SWITCH            (B2) = 0, FINISH
  00005    51  1  00001              INI       1,1
           75  0 P00002              SLJ       NEXT
  00006                         TAB     BSS       100
                                        END

   00003 SYMBOLS
```

```
(5.1)     EVAL                                          03/20/67      ED     0      PAGE NO.     1

                                          IDENT    EVAL          VOLUME II-PROBLEM 5
PROGRAM LENGTH                00020
ENTRY POINTS    EVAL          00003
                                          ENTRY    EVAL
  00000                             DIVIS   BSS      1
  00001   00  0  00000            CON1    DEC      4
          00  0  00004
  00002   00  0  00000            CON2    DEC      3
          00  0  00003
  00003                             EVAL    BSS      1
  00004   12  0 P00016                    LDA      Y
          24  0 P00001                    MUI      CON1
  00005   15  0 P00015                    SUB      X              4Y-X
          20  0 P00000                    STA      DIVIS
  00006   12  0 P00015                    LDA      X
          24  0 P00002                    MUI      CON2
  00007   15  0 P00016                    SUB      Y
          15  0 P00016                    SUB      Y              3X-2Y
  00010   22  2 P00012                    AJP,PL   EXTZER         EXTEND
          04  0  77777                    ENQ      -0             SIGN
  00011   75  0 P00013                    SLJ      DIV            THROUGH
          50  0  00000
  00012   04  0  00000            EXTZER  ENQ      0              Q
          50  0  00000
  00013   25  0 P00000            DIV     DVI      DIVIS
          20  0 P00017                    STA      Z
  00014   75  0 P00003                    SLJ      EVAL
          50  0  00000
  00015                             X       BSS      1
  00016                             Y       BSS      1
  00017                             Z       BSS      1
                                          END
   00011 SYMBOLS
```

```
(5.1)     SOLVE                                         03/20/67      ED     0      PAGE NO.     1

                                          IDENT    SOLVE         VOLUME II-PROBLEM 6
PROGRAM LENGTH                00016
ENTRY POINTS    SOLVE         00005
                                          ENTRY    SOLVE
  00000   77  7  77777            CON1    DEC      -4
          77  7  77773
  00001   00  0  00000            CON2    DEC      135
          00  0  00207
  00002   77  7  77777            X       DEC      -10
          77  7  77765
  00003                             SAVE1   BSS      1              X CUBED
  00004                             SAVE2   BSS      1
  00005                             SOLVE   BSS      1
  00006   12  0 P00002                    LDA      X
          50  0  00000
  00007   24  0 P00002            TRYAGN  MUI      X              X SQUARED
          24  0 P00002                    MUI      X              X CUBED
  00010   20  0 P00003                    STA      SAVE1
          24  0 P00002                    MUI      X              X 4TH
  00011   24  0 P00002                    MUI      X              X 5TH
          20  0 P00004                    STA      SAVE2
  00012   12  0 P00003                    LDA      SAVE1
          24  0 P00000                    MUI      CON1           -4X CUBED
  00013   14  0 P00004                    ADD      SAVE2
          14  0 P00001                    ADD      CON2           X**5-4X**3+135
  00014   22  0 P00005                    AJP,ZR   SOLVE          EXIT
          72  0 P00002                    RAO      X
  00015   75  0 P00007                    SLJ      TRYAGN
          50  0  00000
                                          END
   00007 SYMBOLS
```

```
(5.1)   SWITCH                                          03/20/67      ED    0       PAGE NO.      1

                                        IDENT    SWITCH          VOLUME II-PROBLEM 7
      PROGRAM LENGTH            00014
      ENTRY POINTS    SWITCH    00001
                                        ENTRY    SWITCH
        00000                   C       BSS      1
        00001                   SWITCH  BSS      1
        00002    75  1 P00004           SLJ      SUM             JUMP SW 1 ON
                 75  2 P00006           SLJ      DIFF            JUMP SW 2 ON
        00003    75  3 P00010           SLJ      PROD            JUMP SW 3 ON
                 75  0 P00001           SLJ      SWITCH
        00004    12  0 P00012  SUM      LDA      A
                 14  0 P00013           ADD      B               A+B
        00005    20  0 P00000           STA      C
                 75  0 P00001           SLJ      SWITCH
        00006    12  0 P00012  DIFF     LDA      A
                 15  0 P00013           SUB      B               A-B
        00007    20  0 P00000           STA      C
                 75  0 P00001           SLJ      SWITCH
        00010    12  0 P00012  PROD     LDA      A
                 24  0 P00013           MUI      B               AB
        00011    20  0 P00000           STA      C
                 75  0 P00001           SLJ      SWITCH
        00012                   A       BSS      1
        00013                   B       BSS      1
                                        END

         00007 SYMBOLS
```

```
(5.1)   MAXIMIZE                                        03/20/67      ED    0       PAGE NO.      1

                                        IDENT    MAXIMIZE        VOLUME II-PROBLEM 8
      PROGRAM LENGTH            00023
      ENTRY POINTS    MAXIMIZE  00001
                                        ENTRY    MAXIMIZE
        00000                   MAXF    BSS      1
        00001                   MAXIMIZE BSS     1



        00002    50  0  00000
                 75  4 P00004   *      RTJ      SUBR



        00003    75  0 P00001           SLJ      MAXIMIZE
                 50  0  00000
        00004    75  0  77777  SUBR     SLJ      **              ENTRANCE TO SUBROUTINE
                 12  0 P00020           LDA      A
        00005    15  0 P00021           SUB      B
                 22  2 P00014           AJP,PL   AANDC           A.GT.B
        00006    12  0 P00021           LDA      B
                 15  0 P00022           SUB      C
        00007    22  2 P00012           AJP,PL   TRANSB          B.GT.C
                 50  0  00000
        00010    12  0 P00022  TRANSC    LDA      C
                 20  0 P00000           STA      MAXF
        00011    75  0 P00004           SLJ      SUBR            EXIT SUBROUTINE
                 50  0  00000
        00012    12  0 P00021  TRANSB    LDA      B
                 20  0 P00000           STA      MAXF
        00013    75  0 P00004           SLJ      SUBR            EXIT SUBROUTINE
                 50  0  00000
        00014    12  0 P00020  AANDC     LDA      A
                 15  0 P00022           SUB      C
        00015    22  2 P00016           AJP,PL   TRANSA          A.GT.C
                 75  0 P00010           SLJ      TRANSC
        00016    12  0 P00020  TRANSA    LDA      A
                 20  0 P00000           STA      MAXF
        00017    75  0 P00004           SLJ      SUBR            EXIT SUBROUTINE
                 50  0  00000
        00020                   A       BSS      1
        00021                   B       BSS      1
        00022                   C       BSS      1
                                        END

         00012 SYMBOLS
```

```
(5.1)  SEPARATE                                          03/20/67      ED    0       PAGE NO.      1

                                          IDENT     SEPARATE          VOLUME II-PROBLEM 9
PROGRAM LENGTH              00322
ENTRY POINTS    SEPARATE    00144
                                          ENTRY     SEPARATE
  00000                     RANGE     BSS     100
  00144                     SEPARATE  BSS     1
  00145   50  1  00000               ENI     0,1
          50  2  00000               ENI     0,2
  00146   12  1 P00156   NEXTWORD    LDA     FLUX,1
          50  0  00000
  00147   62  0  77772               RGJP,GT A,-5,LT           NUM.GT.-5
          11  3 P00152
  00150   54  1  00143   CKCNT       ISK     99,1              NO,.LE.
          75  0 P00146               SLJ     NEXTWORD
  00151   75  0 P00144               SLJ     SEPARATE
          50  0  00000
  00152   62  0  00005   LT          RGJP,LT A,5,ENT           NUM.LT.5
          21  3 P00154
  00153   75  0 P00150               SLJ     CKCNT             NO,.GE.
          50  0  00000
  00154   20  2 P00000   ENT         STA     RANGE,2
          51  2  00001               INI     1,2
  00155   75  0 P00150               SLJ     CKCNT
          50  0  00000
  00156                     FLUX      BSS     100
                                      END
   00007 SYMBOLS
```

```
(5.1)      PASS                                          03/20/67      ED    0       PAGE NO.      1

                                          IDENT     PASS              VOLUME II-PROBLEM 10
PROGRAM LENGTH              00012
ENTRY POINTS    PASS        00001
                                          ENTRY     PASS
  00000                     Y         BSS     1
  00001                     PASS      BSS     1



  00002   63  1  00000   READY       BJSX    SUB,1
          04  0 P00005
  00003   00  0  00000               DEC     3,256943
          00  0  00003
  00004   00  0  00000               256943
          00  7  65657



  00005                     SUB       BSS     1
  00006   12  0 P00011               LDA     X
          24  1  00001               MUI     1,1               ADDRESS READY +1
  00007   15  1  00002               SUB     2,1               ADDRESS READY +2
          20  0 P00000               STA     Y
  00010   51  1  00003               INI     3,1               BYPASS DATA
          75  0 P00005               SLJ     SUB
  00011                     X         BSS     1
                                      END
   00005 SYMBOLS
```

```
(5,1)   ODDNUM                                          03/20/67        ED    0         PAGE NO,       1

                                        IDENT     ODDNUM              VOLUME II-PROBLEM 11
     PROGRAM LENGTH              00012
     ENTRY POINTS    ODDNUM      00002
                                        ENTRY     ODDNUM
       00000   00  0  00000  ODDITY   DEC       0
               00  0  00000
       00001   00  0  00000  FIVE      DEC       5
               00  0  00005
       00002                  ODDNUM   BSS       1
       00003   10  0  00001           ENA       1
               00  7  40555           RXT       A,D
       00004   50  1  00002           ENI       2,1
               50  0  00000
       00005   00  5  32053  NEXT      ROP,+     0,B1,A              NEXT ODD
               50  0  00000
       00006   62  0  01750           RGJP,GT   A,1000,ODDNUM
               11  3 P00002
       00007   04  0  00000           ENQ       0
               25  0 P00001           DVI       FIVE
       00010   23  1 P00005           QJP,NZ    NEXT
               72  0 P00000           RAO       ODDITY
       00011   75  0 P00005           SLJ       NEXT
               50  0  00000
                                        END
        00004 SYMBOLS
```

```
(5,1)   REVERSE                                         03/20/67        ED    0         PAGE NO,       1

                                        IDENT     REVERSE             VOLUME II-PROBLEM 12
     PROGRAM LENGTH              00010
     ENTRY POINTS    REVERSE     00000
                                        ENTRY     REVERSE
       00000                  REVERSE  BSS       1
       00001   50  1  00000           ENI       0,1                 CHARACTER COUNTER
               12  0 P00007           LDA       INPUTREG
       00002   77  1  00200           LRS,EO    48
               03  0  00060
       00003   06  0  00052           QLS       42
               07  0  00006           LLS       6                   H
       00004   06  0  00044  NEXTCHAR QLS       36
               07  0  00006           LLS       6
       00005   54  1  00006           ISK       6,1                 8-2 LEFT
               75  0 P00004           SLJ       NEXTCHAR
       00006   20  0 P00007           STA       INPUTREG
               75  0 P00000           SLJ       REVERSE
       00007                  INPUTREG BSS       1
                                        END
        00003 SYMBOLS
```

```
(5.1)    MINFUN                                          03/20/67        ED     0        PAGE NO.      1

                                          IDENT     MINFUN            VOLUME II-PROBLEM 13
PROGRAM LENGTH              00032
ENTRY POINTS    MINFUN      00001
                                          ENTRY     MINFUN
  00000                   MINIMIZE BSS     1
  00001                   MINFUN   BSS     1
  00002   50  1  00000             ENI     U+1
          10  0  00000             ENA     U
  00003   00  7  40555             RAT     A*D
          04  0  00017             ENQ     17B               MASK IN Q
  00004   06  U  00052             QLS     42
          50  0  00000
  00005   43  1 P00022   NEXT      SSU     TRIAL+1
          02  0  00006             QRS     6
  00006   54  1  00007             ISK     7+1               8 CHAR
          75  0 P00005             SLJ     NEXT
  00007   77  1  00200             LRS+EQ  48
          03  0  00060
  00010   07  0  00006   NEXT1     LLS     6
          00  5  26655             HOP.*   A*D*D             D*NEXT CHAR
  00011   10  0  00000             ENA     U
          50  0  00000
  00012   54  2  00007             ISK     7+2               8 CHAR
          75  0 P00010             SLJ     NEXT1
  00013   04  0  00017             ENQ     17B
          50  0  00000
  00014   45  1 P00022   NEXT2     ADL     TRIAL+1
          50  0  00000
  00015   54  1  00007             ISK     7+1               8 CHAR
          75  0 P00014             SLJ     NEXT2
  00016   00  6  26654             HOP.-   A*D*Q
          23  2 P00020             QJP.PL  DIAGSM
  00017   20  0 P00000             STA     MINIMIZE          A.LT.D
          75  U P00001             SLJ     MINFUN
  00020   00  7  40653   DIAGSM    RAT     D*A
          20  0 P00000             STA     MINIMIZE          D.LT.A
  00021   75  0 P00001             SLJ     MINFUN
          50  0  00000
  00022                   TRIAL    BSS
                                   END
   00007 SYMBOLS
```

```
(5.1)    EVAL                                            03/20/67        ED     0        PAGE NO.      1

                                          IDENT     EVAL              VOLUME II-PROBLEM 14
PROGRAM LENGTH              00020
ENTRY POINTS    EVAL        00005
                                          ENTRY     EVAL
  00000                   Z        BSS     1
  00001                   SAVE     BSS     1
  00002   20  0  24000   CONST    DEC     2.+3.+.05
          00  0  00000
  00003   20  0  26000             DEC     3.
          00  0  00000
  00004   17  7  36314             DEC     .05
          63  1  46315
  00005                   EVAL     BSS     1
  00006   12  0 P00016             LDA     X
          32  0 P00017             FMU     Y
  00007   32  0 P00002             FMU     CONST             2XY
          20  0 P00001             STA     SAVE
  00010   12  0 P00016             LDA     X
          32  0 P00016             FMU     X
  00011   32  0 P00003             FMU     CONST+1           3X**2
          31  0 P00001             FSB     SAVE              3X**2-2XY+Y**2
  00012   20  0 P00001             STA     SAVE
          12  0 P00017             LDA     Y
  00013   32  0 P00017             FMU     Y                 Y**2
          30  0 P00001             FAD     SAVE              3X**2-2XY+Y**2
  00014   32  0 P00004             FMU     CONST+2           FINAL EVALUATION
          20  0 P00000             STA     Z
  00015   75  0 P00005             SLJ     EVAL
          50  0  00000
  00016                   X        BSS     1
  00017                   Y        BSS     1
                                   END
   00006 SYMBOLS
```

```
(5.1)  RESISTOR                                        03/20/67      ED    0       PAGE NO.      1

                                          IDENT     RESISTOR            VOLUME II-PROBLEM 15
PROGRAM LENGTH              01774
ENTRY POINTS    RESISTOR    00014
                                          ENTRY     RESISTOR
  00000   60  0  41463  CONST    DEC       -.05,.05
          14  6  31462
  00001   17  7  36314                     .05
          63  1  46315
  00002                TOLRANCE BSS       10
  00014                RESISTOR BSS       1
  00015   50  4  00000          ENI       0,4                 TOLRANCE POINTER
          50  1  01750          ENI       1000,1              NO OF WORDS
  00016   50  2  00000          ENI       0,2                 M+(82) TS FWA
          50  3  00001          ENI       1,3                 SEQUENTIAL INCREMENTER
  00017   12  0 P00001          LDA       CONST+1             UPPER LIMIT
          16  0 P00000          LDQ       CONST               LOWER LIMIT
  00020   63  0  00000  CONSRCH  SEWL      TESTCASE, RESISTOR
          44  0 P00024
  00021   00  7  40555          RXT       A,D                 SAVE
          12  2 P00023          LDA       TESTCASE-1,2        SATISFIED OPERAND
  00022   20  4 P00002          STA       TOLRANCE,4
          51  4  00001          INI       1,4                 BUMP POINTER
  00023   00  7  40653          RXT       0,A                 RESTORE
          75  0 P00020          SLJ       CONSRCH
  00024                TESTCASE BSS       1000
                                END

   00005 SYMBOLS
```

```
(5.1)  GENERATE                                        03/20/67      ED    0       PAGE NO.      1

                                          IDENT     GENERATE            VOLUME II-PROBLEM 16
PROGRAM LENGTH              01761
ENTRY POINTS    GENERATE    01751
                                          ENTRY     GENERATE
  00000                NUM      BSS       1000
  01750   00  0  02525  SWITCH   OCT       252525252525
          25  2  52525
  01751                GENERATE BSS       1
  01752   50  1  00000          ENI       0,1
          10  0  00001          ENA       1
  01753   37  0 P01750  CKSWTCH  SSM       SWITCH
          75  0 P01760          SLJ       ST
  01754   11  0  00004          INA       4
          20  1 P00000          STA       NUM,1
  01755   11  0  00002          INA       2
          50  0  00000
  01756   54  1  01747  CKCNT    ISK       999,1
          75  0 P01753          SLJ       CKSWTCH
  01757   75  0 P01751          SLJ       GENERATE
          50  0  00000
  01760   20  1 P00000  ST       STA       NUM,1
          75  0 P01756          SLJ       CKCNT
                                END

   00006 SYMBOLS
```

```
(5.1)  PARAMCNT                                          03/20/67      ED    0       PAGE NO.      1

                                        IDENT     PARAMCNT          VOLUME II-PROBLEM 17
PROGRAM LENGTH              00053
ENTRY POINTS    PARAMCNT    00024
                                        ENTRY     PARAMCNT
  00000                     PARAM     BSS       20
  00024                     PARAMCNT  BSS       1
  00025   10  0  00000               ENA       0
          50  1  00000               ENI       0,1
  00026   50  2  00000               ENI       0,2
          50  3  00052               ENI       42,3
  00027   63  2  30006   READCHAR    LBYT,Q0,E6,HI,CL CARD,2,3
          52  2 P00041
  00030   50  3  00052               ENI       42,3              EXECUTED ONLY WHEN FINISHED
          51  2  00001               INI       1,2               WITH A MEMORY WORD
  00031   62  0  00073               RGJP,EQ   *,73B,COMMA       COMMA
          01  4 P00036
  00032   62  0  00060               RGJP,EQ   *,60B,*-3         BLANK, READ NEXT CHARACTER
          01  4 P00027
  00033   62  0  00033               RGJP,EQ   *,33B,PERIOD      PERIOD
          01  4 P00040
  00034   06  0  00052               QLS       42
          07  0  00006               LLS       6                 CHARACTER TO A
  00035   75  0 P00027               SLJ       READCHAR
          50  0  00000
  00036   20  1 P00000   COMMA       STA       PARAM,1           STORE PARAMETER
          51  1  00001               INI       1,1
  00037   10  0  00000               ENA       0
          75  0 P00027               SLJ       READCHAR
  00040   20  1 P00000   PERIOD      STA       PARAM,1
          75  0 P00024               SLJ       PARAMCNT
  00041                     CARD      BSS       10
                                      END
   00006 SYMBOLS
```

```
(5.1)  READINT                                           03/20/67      ED    0       PAGE NO.      1

                                        IDENT     READINT           VOLUME II-PROBLEM 18
PROGRAM LENGTH              00326
ENTRY POINTS    READINT     00313
                                        ENTRY     READINT
  00000                     REC1      BSS       100
  00144                     REC3      BSS       100
  00310   70  0  00144     CWA1      IOTR,C    REC1,100
          00  0 P00000
  00311   60  0  00144               IOSR,C    0,100
          00  0  00000
  00312   30  0  00144               IOTR      REC3,100
          00  0 P00144
  00313                     READINT   BSS       1
  00314   74  5  00000               CLCH      0
          00  0  00000
  00315   74  0 P00315               CONN      0,3,2,*
          00  0  03002
  00316   74  1 P00316               EXTF      0,2,*             SET FORMAT TO BCD
          00  0  00002
  00317   74  1 P00317               EXTF      0,3,*             SET DENSITY TO 556
          00  0  00003
  00320   74  1 P00320               EXTF      0,10B,*           REWIND
          00  0  00010
  00321   74  1 P00321               EXTF      0,24B,*           ABNORMAL END OF OPERATION
          00  0  00024
  00322   74  2 P00322               BEGR      0,CWA1,*          TRANSFER DATA
          00  0 P00310
  00323   74  4  11000               COPY      0,1               STATUS TO B1
          00  0  00000
  00324   63  0  02014               NBJP      B1,12,*-1         LOOP IF STILL READING
          60  0 P00323
  00325   75  0 P00313               SLJ       READINT           FINISHED
          50  0  00000
                                      END
   00004 SYMBOLS
```

```
(5.1)     EVAL                                          03/20/67      ED     0       PAGE NO.      1

                                        IDENT     EVAL              VOLUME II-PROBLEM 19
     PROGRAM LENGTH              00015
     ENTRY POINTS    EVAL        00003
                                        ENTRY     EVAL
       00000                X           BSS       1
       00001                Y           BSS       1
       00002                Z           BSS       1
       00003                EVAL        BSS       1
       00004   12  0 P00001             LDA       Y
               32  0 P00011             FMU       =D6.7
       00005   20  0 P00014             STA       =SHOLD
               12  0 P00000             LDA       X
       00006   32  0 P00012             FMU       =D5.8
               31  0 P00014             FSB       =SHOLD          5.8X-6.7Y
       00007   33  0 P00013             FDV       =D-4.9
               20  0 P00002             STA       Z
       00010   75  0 P00003             SLJ       EVAL
               50  0  00000
       00011   20  0  36546
               31  4  63146
       00012   20  0  35631
               46  3  14632
       00013   57  7  43063
               14  6  31462
                                        END
        00005 SYMBOLS
```

```
(5.1)     A                                             03/20/67      ED     0       PAGE NO.      1

                                        IDENT     A                 VOLUME II-PROBLEM 20
     PROGRAM LENGTH              00010
     ENTRY POINTS    A           00001
     EXTERNAL SYMBOLS
                     ELEMENTS
                                        ENTRY     A
                                        EXT       ELEMENTS
       00000                SUM         BSS       1
       00001                A           BSS       1
       00002   50  1  00000             ENI       0,1
               50  0  00000
       00003   77  1  04000             LDA       ($)ELEMENTS,1
               12  1 X77777
       00004   14  1 X77777  NEXT      ADD       ELEMENTS+1,1
               50  0  00000
       00005   54  1  00010             ISK       0,1
               75  0 P00004             SLJ       NEXT
       00006   77  1  04000             STA       (*)SUM          IN THIS BANK
               20  0 P00000
       00007   75  0 P00001             SLJ       A
               50  0  00000
                                        END
        00004 SYMBOLS
```

```
(5.1)          A                                                        03/20/67         ED      0         PAGE NO.        1
                                               IDENT      A                       VOLUME II-PROBLEM 21
      PROGRAM LENGTH               00012
      ENTRY POINTS      A          00004
                        X          00000
                        Y          00001
                        Z          00002
      EXTERNAL SYMBOLS
                        M
                        B
                                               ENTRY      A,X,Y,Z
                                               EXT        M
        00000                     X            BSS        1                       ASSUME FLOATING POINT
        00001                     Y            BSS        1                       NUMBERS PRESTORED
        00002                     Z            BSS        1
        00003                     R            BSS        1
        00004                     A            BSS        1
        00005                                  CALL       B                       GO TO (*)B. SWITCH OR TO THIS
        00006    32  0 P00011                  FMU        =D5.                    BANK
                 50  0  00000
        00007    77  1  04000                  FMU        ($)M                    EXTERNAL
                 32  0 X77777
        00010    20  0 P00003                  STA        R                       R=5MZ
                 75  0 P00004                  SLJ        A
        00011    20  0  35000
                 00  0  00000
                                               END
         00007 SYMBOLS
```

```
(5.1)          B                                                        03/20/67         ED      0         PAGE NO.        1
                                               IDENT      B
      PROGRAM LENGTH               00013
      ENTRY POINTS      B          00001
                        M          00000
      EXTERNAL SYMBOLS
                        X
                        Y
                        Z
                                               ENTRY      B,M
                                               EXT        X,Y,Z
        00000                     M            BSS        1
        00001                     B            BSS        1
        00002    12  0 X77777                  LDA        X
                 50  0  00000
        00003    77  1  04000                  FMU        (*)=D3.
                 32  0 P00011
        00004    77  1  04000                  FSB        ($)Y
                 31  0 X77777
        00005    31  0 X77777                  FSB        Y                       3X-2Y
                 50  0  00000
        00006    77  1  04000                  FDV        (*)=D7.
                 33  0 P00012
        00007    77  1  04000                  STA        ($)Z                    EXTERNAL
                 20  0 X77777
        00010    75  0 P00001                  SLJ        B
                 50  0  00000
        00011    20  0  26000
                 00  0  00000
        00012    20  0  37000
                 00  0  00000
                                               END
         00005 SYMBOLS
```

```
(5.1)        A                                          03/20/67        ED     0        PAGE NO.       1

                                        IDENT     A                    VOLUME II-PROBLEM 22
      PROGRAM LENGTH              00005
      ENTRY POINTS     A          00000
      BLOCK NAMES
                       B1         00005
                                  ENTRY     A
  00000                      B1   BLOCK     5
  00000                           COMMON    P,R,S,T,CON1
  00001                                     R
  00002                                     S
  00003                                     T
  00004                                     CON1
                     C00004       ORGR      CON1
  00004   20  0  04000            DEC       .5
          00  0  00000
                     P00000       ORGR      *
  00000                      A    BSS       1
  00001   77  1  04000            LDA       ($)S
          12  0  C00002
  00002   30  0  C00003            FAD       T
          32  0  C00001            FMU       R
  00003   32  0  C00004            FMU       CON1
          20  0  C00000            STA       P
  00004   75  0  P00000            SLJ       A
          50  0  00000
                                  END
   00006 SYMBOLS
```

```
(5.1)    MATMUL                                         03/20/67        ED     0        PAGE NO.       1

                                        IDENT     MATMUL               VOLUME II-PROBLEM 23
      PROGRAM LENGTH              00020
      ENTRY POINTS     MATMUL     00012
                       RETURN     MACRO     (P1,P2)
                                  LDA       A+P1+3*P2-4              CONVERTED ADDRESS
                                  ENDM
                                  ENTRY     MATMUL
  00000                      Z    BSS       1
  00001                      A    BSS       9                        ARRAY
  00012                      MATMUL BSS       1
  00013                           RETURN    (1,1)
          20  0  P00000            STA       Z
  00014                           RETURN    (2,2)
          32  0  P00000            FMU       Z
  00015   20  0  P00000            STA       Z
                                  RETURN    (3,3)
  00016   32  0  P00000            FMU       Z
          20  0  P00000            STA       Z
  00017   75  0  P00012            SLJ       MATMUL
          50  0  00000
                                  END
   00003 SYMBOLS
```

```
(5.1)     MODE                                          03/20/67        ED     0        PAGE NO.       1

                                        IDENT     MODE                VOLUME II-PROBLEM 24
     PROGRAM LENGTH              00004
     ENTRY POINTS    MODE        00000
                          OPMODE  MACRO     (P1,P2)
                                  IFT,EQ    P1,/BCD/,1
                                  EXTF      1,2,*             SET TO BCD
                                  IFT,EQ    P1,/BIN/,1
                                  EXTF      1,1,*             SET TO BIN
                                  IFT,EQ    P2,/200/,1
                                  EXTF      1,4,*             200 BPI
                                  IFT,EQ    P2,/556/,1
                                  EXTF      1,3,*             556BPI
                                  IFT,EQ    P2,/800/,1
                                  EXTF      1,6,*             800 BPI
                                  ENDM
                                  ENTRY     MODE
     00000                MODE    BSS       1
     00001                        OPMODE    (BIN,556)
     00003   75  0 P00000         SLJ       MODE
             50  0  00000
                                  END
      00001 SYMBOLS
```

```
(5.1)     NUT GOODY                                     03/20/67        ED     0        PAGE NO.       1

                                        IDENT     ILLUST              VOLUME II-PROBLEM 25
     PROGRAM LENGTH              00005
     ENTRY POINTS    ILLUST      00000
                                  ENTRY     ILLUST
                                            THIS IS A DO NOTHING
                                            PROGRAM FOR ILLUSTRATION ONLY.



                                            WE HAVE JUST SPACED 5 LINES
                                            AND WE SPACE 5 MORE IN
                                            ORDER TO SEPARATE THIS AREA.



                                            NOW LET'S ENTITLE THIS PAGE.
                                            YOU SHOULD SEE NUT GOODY
                                            AT THE TOP.
                                            NOW LET'S START
                                            OUR PROGRAM.
     00000                ILLUST  BSS       1
     00001   12  0 P00003         LDA       CON1
             30  0 P00004         FAD       CON2
     00002   75  0 P00000         SLJ       ILLUST



             50  0  00000
                                            NOW LET'S ASSEMBLE
                                            THE DECIMAL CONSTANTS
                                            BUT NOT LIST THEM.
                                            THE CONSTANTS ARE
                                            ACTUALLY ASSEMBLED.
                                            NO ASSEMBLY ERROR
                                            APPEARS.



                                            THANKS FOR THE SHOW
                                            AND NOW WE SAY,
                          THE
                                  END
      00003 SYMBOLS
```

```
 BEGIN JOB AT   1018 = 41
J08,54587,DJE,3
SCOPE VERSION   6.28*

COMPASS,L,X
```

```
(5.1)   COMPARE                                          03/20/67        ED      0          PAGE NO.        1

                                          IDENT     COMPARE
      PROGRAM LENGTH              00050
      ENTRY POINTS     COMPARE    00016
      EXTERNAL SYMBOLS
                SENTRY
                                          ENTRY     COMPARE
 00000    00  6  06060  OUTPUT    BCD       1,0                 TO DOUBLE SPACE AND INDENT LIST
          60  6  06060
 00001                   INPUT     BSS       10
 00013    10  0  00012  CWA1      IOTW      INPUT,10
          00  0 P00001
 00014    10  0  00013  CWA2      IOTW      OUTPUT,11           TO INDENT LIST
          00  0 P00000
 00015    01  0  50000  CON1      BCD       1,15000
          00  6  06060
 00016                   COMPARE   BSS       1
 00017                             MODE      (60,*,BCD)          SET MODE TO BCD
 00022                   NEXT      READ      (60,CWA1,*)         READ ONE CARD
 00025                   NOTFIN    STATUS    (60)
 00027    23  3 P00025             UJP,MI    NOTFIN              UNIT STILL IN OPERATION
          50  0  00000
 00030    16  0 P00001  INPFIN    LDQ       INPUT
          10  0  00000             ENA       0
 00031    07  0  00014             LLS       12
          11  0  77272             INA       -5058               BCD 55
 00032    22  0 P00047             AJP,ZR    FINISHED            DATA CARD TERMINATOR
          50  1  00036             ENI       30,1                USED TO ALLIGN SAL. WITH 15,000
 00033    16  0 P00006             LDQ       INPUT+5
          50  0  00000
 00034    10  0  00000  NEXTCHAR   ENA       0
          07  0  00006             LLS       6
 00035    62  0  00060             RGJP,EQ   A,60B,ALLIGN        A=BLANK
          01  3 P00037
 00036    51  1  77771             INI       -6,1                DECREASE SHIFT COUNTER
          75  0 P00034             UJP       NEXTCHAR
 00037    12  0 P00006  ALLIGN     LDA       INPUT+5
          01  1  00000             ARS       1                   ALLIGN WITH $15,000
 00040    15  0 P00015             SUB       CON1                $15,000
          22  3 P00022             AJP,MI    NEXT                BELOW $15,000
 00041                             WRITE     (61,CWA2,*)         AT OR ABOVE $15,000
 00044                   WAIT      STATUS    (61)
 00046    23  3 P00044             UJP,MI    WAIT                WAIT
          75  0 P00022             SLJ       NEXT
 00047                   FINISHED  EXIT
                                   END       COMPARE

  00016 SYMBOLS
```

```
LOAD
RUN,5,40,7
```

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN, 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

Pub. No. 60166700 Rev. B

Litho in U.S.A.