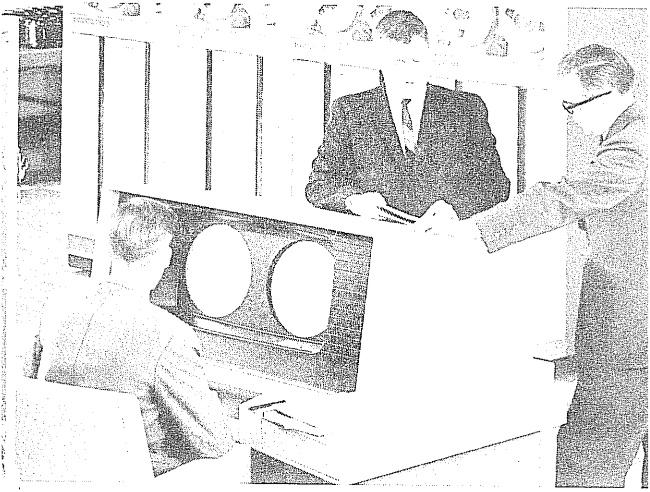
DAVID E. LEE
2.9.76

[illegible]



INSTANT

2·3

FORTRAN

FORTRAN 2.3 provides a scientific language for programming under control of the SCOPE Operating System on Control Data 6000 series computers. FORTRAN is compatible with FORTRAN II, FORTRAN IV and FORTRAN VI languages.

Expressions and Statements

Arithmetic, logical, relational, and masking operations may be specified for the evaluation of expressions. Arithmetic expressions may contain any combination of arithmetic modes. Relational expressions contain arithmetic expressions separated by relational operators. Logical expressions may contain logical variables, logical constants, and relational expressions separated by logical operators. The masking expression is a generalized form of the logical expression in which the variables are of types other than logical.

Replacement statements may be arithmetic, logical and masking.

Variable and Function Types

Variable and function types may be implied as real or integer by the initial character or explicitly defined by a Type declaration as

COMPLEX
REAL
DOUBLE or DOUBLE PRECISION
INTEGER
LOGICAL

Control Statements

Control statements alter sequential execution of statements, perform tests and iterations, and terminate subprograms.

GO TO
DO
IF
CONTINUE
STOP
PAUSE
END

Input/Output

Input/output statements permit the programmer to make use of computer efficiency with a complete range of input/output formats.

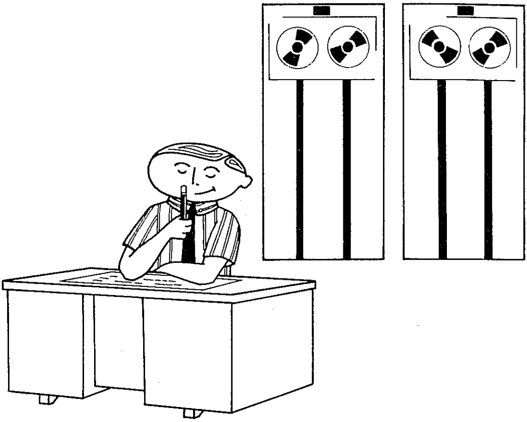
PRINT
PUNCH
BUFFER
WRITE
READ

Functions and Subprograms

Statements and statement sequences may be written as statement functions and as main programs, subroutines, and functions.

Library

The FORTRAN library provides a full set of mathematical and utility functions and subroutines.



FORTRAN PROGRAM EFFICIENCY HINTS

Same mode variables and constants in an arithmetic expression

Reduced use of subscripts

Constant subscripts rather than variable

Arguments in common rather than calling list

Non-varying values computed before entering DO loop

Avoid function names as parameters in subprogram calling sequences

Limited use of logical mode

FORTRAN ELEMENTS

Constants	Form	Examples
Integer	$n_1 n_2 \ldots n_m$ $1 \leq m \leq 18$	2 247 314159265
Octal	$On_1 \ldots n_m$ $6 \leq m \leq 20$ $n_1 \ldots n_m B$	7777777700000000B O23146541 O000077
Real	$n_1 n_2 \ldots n_m \quad E \pm exp_{10}$ $1 \leq m \leq 15$	3.14 .0749 314.E05 .3E01
Double	$n_1 n_2 \ldots n_m \quad D \pm exp_{10}$ $1 \leq m \leq 18$	37986324.3201D+01 3.1415D0
Complex	(r_1, r_2) real numbers r_1 = real part, r_2 = imaginary part	(1.,6.55) (-15.,16.7) (0.,-1.)
Hollerith	$nHf_1 f_2 \ldots f_n$ $1 \leq n \leq 10$ for a replacement statement $1 \leq n \leq 136$ for a format statement $1 \leq n \leq 1307$ for a DATA statement	7HGOGETIT 3HSON
	f_i alphanumeric character	5HTHANX
Logical	.TRUE. .T.	all one bits
	or stored as	
	.FALSE. .F.	all zero bits

Subscripts

For DIMENSION A(L,M,N) the location of A(i,j,k) with respect to first element of A is

A+(i-1+L*(j-1+M*(k-1)))*E

E is the number of words occupied by each element of A.

A subscript may be:

- Integer constant
- Simple integer variable
- Simple integer arithmetic expression

Examples:

(I,J)
(I+3, J+3, 2*K+1)
(3*K*I+3)

Variables	Form	Examples
Simple integer	$a_1a_2...a_m$ $1 \le m \le 7$ a_1 I,J,K,L,M,or N a_2 - a_7 Alphanumeric	N I2S04 M58
Simple real	$a_1a_2...a_m$ $1 \le m \le 7$ a_1 Alphabetic other than I,J,K,L,M,or N a_2 - a_7 Alphanumeric	VECTOR SPOILS A65
Subscripted integer	$a_1a_2...a_m$ (i,j,k) $1 \le m \le 7$ a_1 I,J,K,L,M,or N a_2 - a_7 Alphanumeric	NERVE (6,8,6) LO (J) JEL (I,M,3)
Subscripted real	$a_1a_2...a_m$ (i,j,k) $1 \le m \le 7$ a_1 Alphabetic other than I,J,K,L,M,or N a_2 - a_7 Alphanumeric	TIMIE (J,K,L) QL (1) ROGER (2,2,1)

Variables defined by Type declarations begin with any letter.

FORTRAN STATEMENTS

A subprogram is compiled in one of two modes:

FORTRAN IV

FORTRAN II

Subprogram Statements

PROGRAM name $(f_1,\ldots,f_n)$

FORTRAN VI PROGRAM name $(f_1,\ldots,f_n)$

FORTRAN IV PROGRAM name $(f_1,\ldots,f_n)$

FORTRAN II PROGRAM name $(f_1,\ldots,f_n)$

SUBROUTINE name $(p_1,\ldots,p_m)$

FORTRAN VI SUBROUTINE name $(p_1,\ldots,p_m)$

FORTRAN IV SUBROUTINE name $(p_1,\ldots,p_m)$

FORTRAN II SUBROUTINE name $(p_1,\ldots,p_m)$

CALL name $(p_1,\ldots,p_m)$

$m \le 60 \quad n \le 50$

Parameter list may be omitted

FUNCTION name $(p_1,\ldots,p_m)$

type FUNCTION name $(p_1,\ldots,p_m)$

FORTRAN VI FUNCTION name $(p_1,\ldots,p_m)$

FORTRAN VI type FUNCTION name $(p_1,\ldots,p_m)$

FORTRAN IV FUNCTION name $(p_1,\ldots,p_m)$

FORTRAN IV type FUNCTION name $(p_1,\ldots,p_m)$

FORTRAN II FUNCTION name $(p_1,\ldots,p_m)$

FORTRAN II type FUNCTION name $(p_1,\ldots,p_m)$

BLOCK DATA

EXTERNAL name_1, $\text{name}_2,\ldots$

RETURN

ENTRY name

Data Declaration and Storage Allocation Statements

COMPLEX list
DOUBLE list
DOUBLE PRECISION list
REAL list
INTEGER list
LOGICAL list
DIMENSION $v_1,v_2,...,v_n$
COMMON/l_1/list$_1$/l_2/list$_2$...
EQUIVALENCE $(a_1,b_1,...),(a_2,b_2,...),...$
DATA list$_1$/$a_1,...,a_n$/,list$_2$/$b_1,...,b_n$/,...or
DATA (list$_3$=$c_1,...,c_n$),(list$_4$=$d_1,...,d_n$),...

Statement Function

name $(p_1,...,p_n)$=expression

Replacement Statements

a = Arithmetic expression
l = Logical expression
m = Masking expression

Control Statements

GO TO n
GO TO i,$(n_1,...,n_i)$
GO TO i
ASSIGN n to i
GO TO $(n_1,...,n_i)$,i
IF (a) n_1,n_2,n_3
IF (l) s
IF (l) n_1,n_2

DO n i=m_1,m_2,m_3
CONTINUE
PAUSE
PAUSE n
STOP
STOP n
END

Format Statement and Specifications

FORMAT (spec$_1$,...,spec$_n$)
Spec$_i$

Ew.d	Single precision floating point with exponent
Fw.d	Single precision floating point without exponent
Dw.d	Double precision floating point with exponent

Iw	Decimal integer
Rw	Alphanumeric, right justified, leading zeros
Ow	Octal integer
Aw	Alphanumeric, left justified, with trailing blanks
Lw	Logical
nP	Scaling factor

Complex values are converted by a pair of consecutive Ew.d or Fw.d.

wX	Intra-line spacing
wH *...*	Heading and labeling, Hollerith characters
/	Begin new record

Printer Carriage Control

Character in first column	Resulting PRINT Operation
0	Double space before printing
1	Eject page before printing
+	Suppress spacing before printing
blank or other than above	Single space before printing

Input/Output and Data Transmission

READ (i,n) list

    READ INPUT TAPE i,n,list

WRITE (i,n) list

    WRITE OUTPUT TAPE i,n,list

READ (i) list

    READ TAPE i,list

WRITE (i) list

    WRITE TAPE i,list

IF (EOF,i) n_1,n_2

IF (ENDFILE i) n_1,n_2

END FILE i

REWIND i

BACKSPACE i

BUFFER IN (i,m) list

BUFFER OUT (i,m) list

IF (UNIT, i) n_1,n_2,n_3,n_4

n_1	Busy	n_3	EOF
n_2	Complete	n_4	Parity error

NAMELIST $/y_1/a_1/y_2/a_2/\ldots/y_n/a_n$

READ n, list

PRINT n, list

PUNCH n, list

ENCODE (c,n,v) L

DECODE (c,n,v) L

FORTRAN FUNCTIONS

In-Line Functions

ABS(x)	Absolute value	Real to real
AIMAG(c)	Imaginary part of complex	Complex to real
AINT(x)	Truncation, integer	Real to real
AMAX0(i_1,i_2,...)	Maximum argument	Integer to real
AMAX1(x_1,x_2,...)	Maximum argument	Real to real
AMIN0(i_1,i_2,...)	Minimum argument	Integer to real
AMIN1(x_1,x_2,...)	Minimum argument	Real to real
AMOD(x_1,x_2)	x_1 modulo x_2	Real to real
AND(x_1,.., x_n)	Boolean AND of x_1,...,x_n	Logical
CMPLX(x_1,x_2)	Real to complex ($x_1 + ix_2$)	Real to complex
COMPL(x)	Complement of x	Logical
CONJG(c)	Conjugate of c	Complex to complex
DIM(x_1,x_2)	If $x_1 > x_2$:x_1-x_2 If $x_1 \leq x_2$:0	Real to real
DMAX1(d_1,d_2,...)	Maximum argument	Double to double
DMIN1(d_1,d_2,...)	Minimum argument	Double to double
FLOAT(i)	Integer to real	Integer to real
IABS(i)	Absolute value	Integer to integer
IDIM(i_1,i_2)	If $i_1 > i_2$:i_1-i_2 If $i_1 \leq i_2$:0	Integer to integer
IFIX(x)	Real to integer	Real to integer

INT(x)	Truncation, integer	Real to integer
ISIGN(i_1,i_2)	Sign of i_2 times $\lvert i_1 \rvert$	Integer to integer
MAX0($i_1,i_2,\ldots$)	Maximum argument	Integer to integer
MAX1($x_1,x_2,\ldots$)	Maximum argument	Real to integer
MIN0($i_1,i_2,\ldots$)	Minimum argument	Integer to integer
MIN1($x_1,x_2,\ldots$)	Minimum argument	Real to real
MOD(i_1,i_2)	i_1 modulo i_2	Integer to integer
OR($x_1,\ldots,x_n$)	Boolean OR of $x_1,\ldots,x_n$	Logical
REAL(c)	Real part of complex	Complex to real
SIGN(x_1,x_2)	Sign of x_2 times $\lvert x_1 \rvert$	Real to real

Library Functions

ACOS(x)	Arccosine	Real to real
ALOG(x)	Natural log of x	Real to real
ALOG10(x)	Log to the base 10 of x	Real to real
ASIN(x)	Arcsine	Real to real
ATAN(x)	Arctangent x radians	Real to real
ATAN2(x_1,x_2)	Arctangent x_1/x_2	Real to real
CABS(c)	Absolute value	Complex to real
CCOS(c)	Complex cosine	Complex to complex
CEXP(c)	Complex exponent	Complex to complex
CLOG(c)	Complex log	Complex to complex
COS(x)	Cosine x radians	Real to real
CSIN(c)	Complex sine	Complex to complex
CSQRT(c)	Complex square root	Complex to complex
DABS(d)	Absolute value	Double to real

DATAN(d)	Double arctangent	Double to double
DATAN2(d_1,d_2)	Double arctangent: d_1/d_2	Double to double
DBLE(x)	Real to double	Real to double
DCOS(d)	Double cosine	Double to double
DEXP(d)	Double exponent	Double to double
DLOG(d)	Natural log of d	Double to double
DLOG10(d)	Log to base 10 of d	Double to double
DMOD(d)	d_1 modulo d_2	Double to double
DSIGN(d_1,d_2)	Sign of: d_2 times d_1 in absolute value	Double to double
DSIN(d)	Sine of double precision argument	Double to double
DSQRT(d)	Square root of double	Double to double
EXP(x)	e to xth power	Real to real
IDINT(d)	Double to integer	Double to integer
LEGVAR(a)	Returns -1 if a indefinite, +1 if a out of range, 0 if a normal	Real to integer
LENGTH(i)	Number of words read on unit i after BUFFER IN	Integer to integer
RANF(i)	Random number generator	Integer to real
SNGL(d)	Double to real (unrounded)	Double to real
SIN(x)	Sine x radians	Real to real
SQRT(x)	Square root of x	Real to real
TAN(x)	Tangent x radians	Real to real
TANH(x)	Hyperbolic tangent x radians	Real to real
LOCF(a)	Address of argument a	Integer
XLOCF(a)	Address of argument a	Integer

LIBRARY SUBROUTINE

DISPLA (nH name, name) DISPLAY NAME AND VALU

$n \leq 7$

DUMP
PDUMP $(a_1,b_1,f_1,\ldots,a_n,b_n,f_n)$ DUMP STORAG

a_i First word of area to be dumped

b_i Last word of area to be dumped

f_i Dump format indicators:

0 or 3 is octal dump

1 is real dump

2 is integer dump

DVCHK (j) DIVISION BY ZERO TES

j=1 if occurred

j=2 if not

FTNBIN(1,n,IRAY)	BINARY BLOCKING OF I/O
READEC(cm,ecs,n)	TRANSFERS WORDS FROM ECS TO CENT MEMORY
WRITEC(cm,ecs,n)	TRANSFERS WORDS FROM CENTRAL MEMORY TO ECS
OPENMS(u,ix,1,p)	OPENS MASS STORAGE FILE
READMS(u,fwa,n,i)	TRANSFERS DATA FROM MASS STORAGE FILE
WRITMS(u,fwa,n,i)	TRANSFERS DATA TO MASS STORAGE FI
STINDX(u,ix,1)	CHANGES THE FILE INDEX

EXIT TERMINATE EXECUTIO

OVERFL (j) OVERFLOW TE

j=1 if occurred

j=2 if not

REMARK (nH . . .) MESSAGE TO SYSTEM DAYFI

Displays it on the console display. nH indicates n hollerith characters in the remark; $n \leq 40$

SECOND(i) TIME IN SECONDS FROM DEADSTART

SLITE (i) TURN ON SENSE LIGHT I

SLITET (i,j) TEST SENSE LIGHT I

j=1 if on

j=2 if off

SSWTCH (i,j) TEST SENSE SWITCH I

j=1 if down

j=2 if up

SEGMENT(fn,e,a,lib,m) LOAD RELOCATABLE SEGMENT

fn Variable name of location containing left-justified display code file name to be loaded

e Segment load level, $00\text{-}77_8$

a Simple or subscripted variable array name containing list of segments, sections, subprograms to be loaded

lib Load unsatisfied externals from system library if zero or blank

m Do not print map of segment load if zero or blank

OVERLAY(fn,l_1,l_2,p) LOAD ABSOLUTE OVERLAY

fn Variable name of location containing left-justified display code file name to be loaded

l_1 Primary overlay level

l_2 Secondary overlay level

p If 6HRECALL, overlay is not reloaded if already in memory

FORTRAN VI DIFFERENCES

END statement in function or subroutine acts as RETURN

DO loop not executed if initial value exceeds terminal value

FORTRAN II AND FORTRAN IV SUBPROGRAMS

Column 1 Indicators

D	Double precision mode replacement
I	Complex mode replacement
B	Masking replacement
F	EXTERNAL

FORTRAN II statements which contain a B in column 1 (Boolean) are evaluated as masking expressions. The operator equivalences are:

FORTRAN	FORTRAN II
.AND.	*
.NOT.	-
.OR.	+
	/

Statements

IF (I)n_1,n_2

If I is a variable, the true branch is taken only if the value of I is negative.

SENSE LIGHT i

i must be integer constant

IF (SENSE LIGHT i)n_1,n_2

i must be integer constant

IF (SENSE SWITCH i)n_1,n_2

i must be integer constant

IF DIVIDE CHECK n_1,n_2

IF QUOTIENT OVERFLOW n_1,n_2

IF ACCUMULATOR OVERFLOW n_1,n_2

FORTRAN II Mode Subprogram Only

EQUIVALENCE may re-order COMMON.

Function names must end with F and be 4-7 characters.

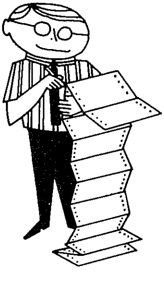
Function names must begin with X if the value is integer and any other alphabetic character if the value is real.

CHARACTER CODES

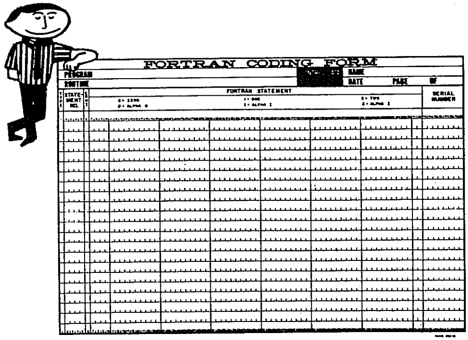
Character	External Code	Console Display Code	Punch Card
A	61	01	12,1
B	62	02	12,2
C	63	03	12,3
D	64	04	12,4
E	65	05	12,5
F	66	06	12,6
G	67	07	12,7
H	70	10	12,8
I	71	11	12,9
J	41	12	11,1
K	42	13	11,2
L	43	14	11,3
M	44	15	11,4
N	45	16	11,5
O	46	17	11,6
P	47	20	11,7
Q	50	21	11,8
R	51	22	11,9
S	22	23	0,2
T	23	24	0,3
U	24	25	0,4
V	25	26	0,5
W	26	27	0,6
X	27	30	0,7
Y	30	31	0,8
Z	31	32	0,9
0	12	33	0
1	01	34	1
2	02	35	2
3	03	36	3
4	04	37	4
5	05	40	5
6	06	41	6
7	07	42	7
8	10	43	8
9	11	44	9
/	21	50	0,1
+	60	45	12
-	40	46	11
blank	20	55	space
.	73	57	12,8,3
)	74	52	12,8,4
$	53	53	11,8,3
*	54	47	11,8,4
,	33	56	0,8,3

(	34	51	0,8,4
=	13	54	8,3
≡	36	60	0,8,6
[	17	61	8,7
]	32	62	0,8,2
:	00	63	8,2
≠	14	64	8,4
→	35	65	0,8,5
∨	52	66	11,0
∧	37	67	0,8,7
↑	55	70	11,8,5
↓	56	71	11,8,6
<	72	72	12,0
>	57	73	11,8,7
≤	15	74	8,5
≥	75	75	12,8,5
¬	76	76	12,8,6

; , character 77, is restricted in FORTRAN



FORTRAN CODING FORM



Each line of a FORTRAN coding form represents the four fields of a punched card.

Field	Columns
Statement Number	1-5
Continuation	6
Statement	7-72
Identification	73-80

Statements may be identified by an integer from 1 through 99999. If a $, C or * appears in column 1, the remainder of the card is ignored by the compiler, but printed with the source listing as a comment.

A $ may be used to separate multiple statements on a card. However, DATA and FORMAT statements cannot appear on a card separated in this manner.

A punch other than zero in column 6 identifies a card as a continuation of the statement from the preceding card.

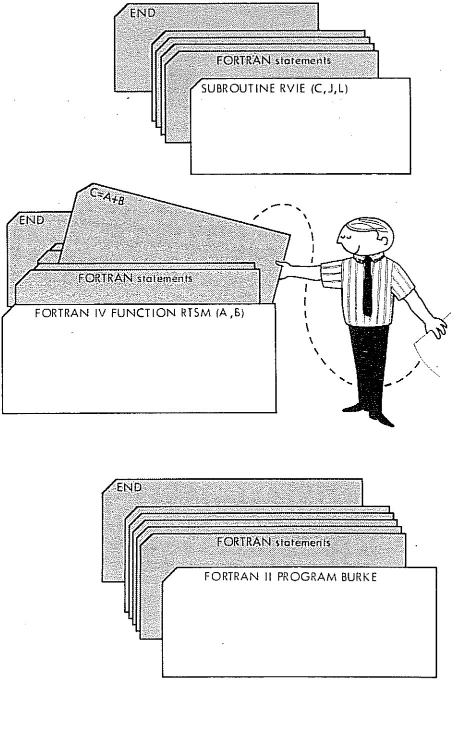
A statement is written in columns 7 through 72; blanks are ignored.

Columns 73-80 may contain identification and serial numbers. These columns are ignored, but printed with the program listing.

The entire 80 columns may be used for data input.

A period in column 1 signals page eject.

FORTRAN SOURCE DECK



FORTRAN CONTROL CARD

RUN(cm,fl,bl,if,of,rf,lc,as,cs)

cm Compiler mode option; if omitted, assume G; if unrecognized, assume S.

G Compile and execute, nolist unless explicit LIST cards appear in the deck

S Compile with source list, no execute

P Compile with source list and punch deck on file PUNCHB, no execute

L Compile with source and object list, no execute

M Compile with source and object list, produce a punch deck on file PUNCHB, no execute

fl Object program field length$_8$; if omitted, it is set equal to the field length at compile time.

bl Object program I/O buffer lengths$_8$; if omitted, assumed to be 2023B.

if File name for compiler input; if omitted assumed to be INPUT.

of File name for compiler output; if omitted, assumed to be OUTPUT.

rf File name on which the binary information is always written; if omitted, assumed to be LGO.

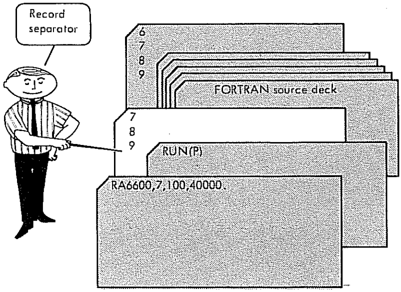
lc Line-limit (octal) on the OUTPUT file of an object program. If omitted, assumed to be 10000_8. If the line count exceeds the specified line limit, the job is terminated.

as If non-zero or non-blank, the ASA switch causes the ASA I/O list/format interaction at execution time. It has no effect on the compilation method.

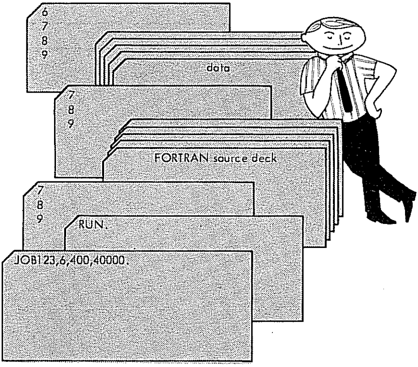
cs Cross-reference switch; if non-zero a cross reference listing is produced.

If parameter list is omitted, RUN must be followed by a period.

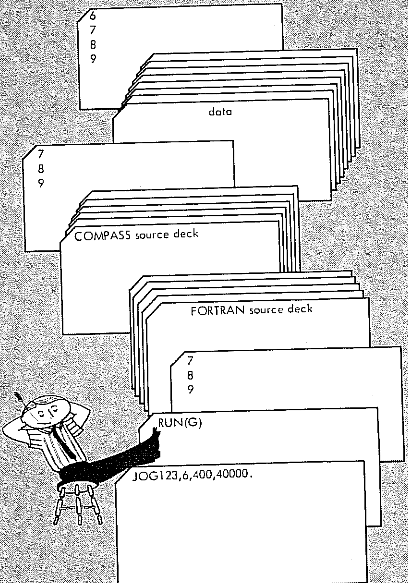
FORTRAN COMPILATION



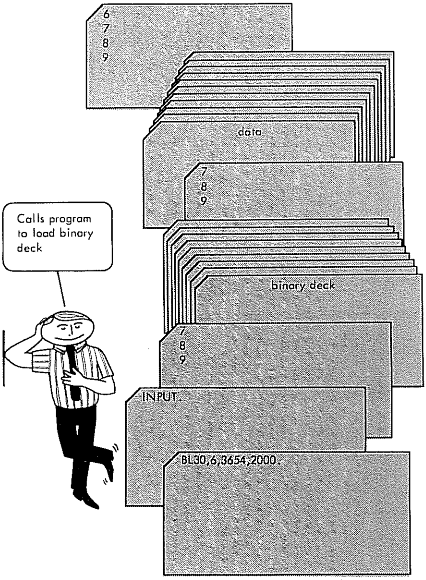
FORTRAN COMPILATION AND EXECUTION



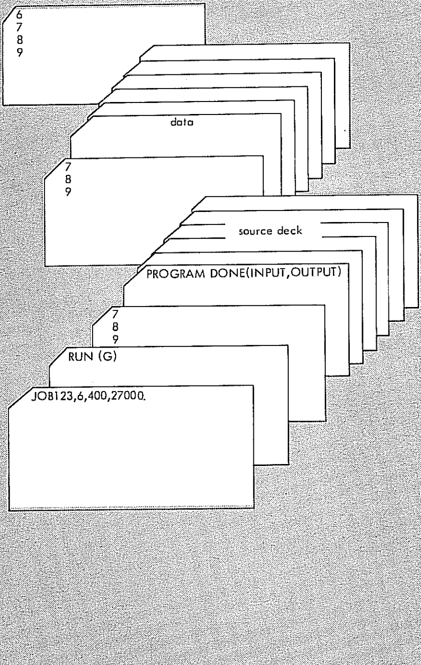
FORTRAN COMPILATION, COMPASS ASSEMBLY AND EXECUTION



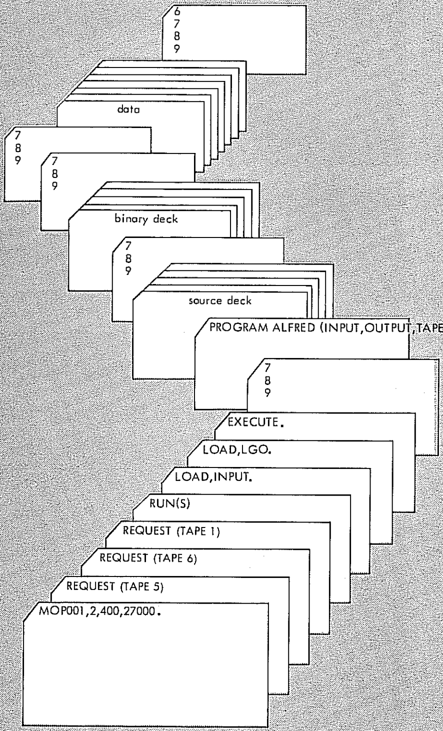
EXECUTION WITH BINARY DECKS



FORTRAN COMPILE AND EXECUTE

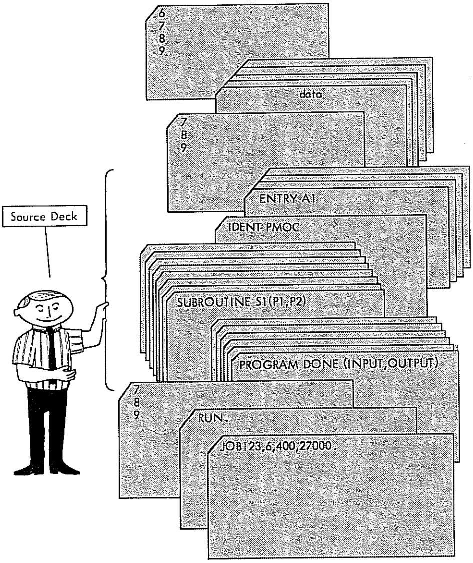


FORTRAN COMPILE AND EXECUTE

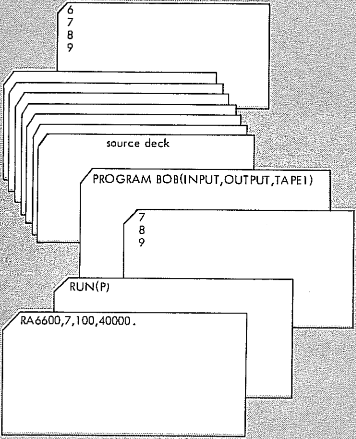


```
MOP001,2,400,27000.
REQUEST (TAPE 5)
REQUEST (TAPE 6)
REQUEST (TAPE 1)
RUN(S)
LOAD,INPUT.
LOAD,LGO.
EXECUTE.
PROGRAM ALFRED (INPUT,OUTPUT,TAPE1,TAPE5,TAPE
```

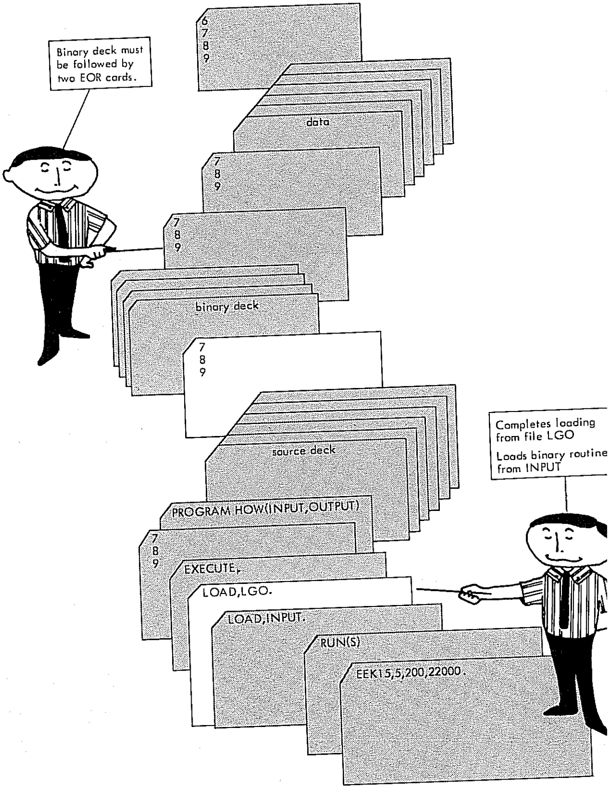
FORTRAN
COMPILE AND EXECUTE WITH MIXED DECK



FORTRAN COMPILE AND PRODUCE BINARY CARDS



FORTRAN COMPILE AND EXECUTE



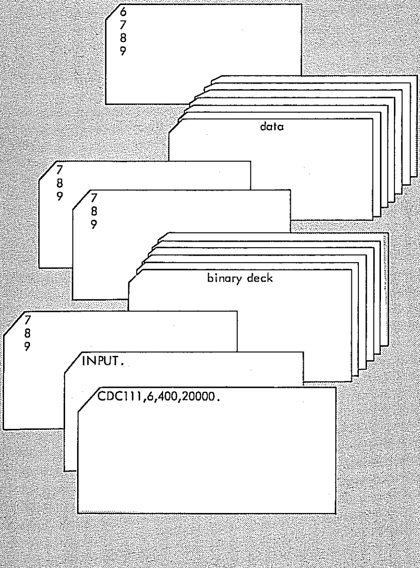
Binary deck must be followed by two EOR cards.

Completes loading from file LGO

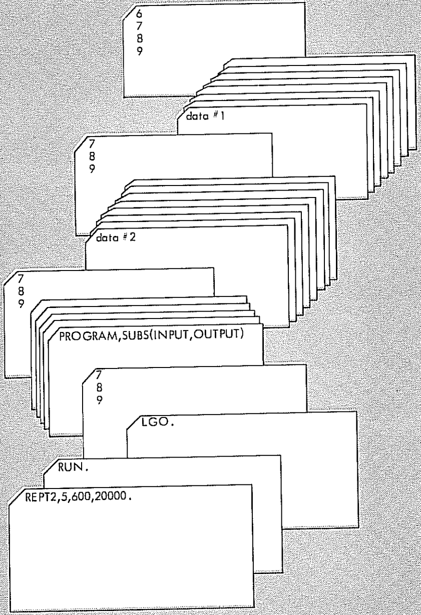
Loads binary routine from INPUT

```
EEK15,5,200,22000.
RUN(S)
LOAD,INPUT.
LOAD,LGO.
EXECUTE.
7
8
9
PROGRAM HOW(INPUT,OUTPUT)
```

LOAD AND EXECUTE BINARY PROGRAM



COMPILE ONCE AND EXECUTE WITH DIFFERENT DATA DECKS



OVERLAY AND SEGMENT CONTROL CARDS

SEGMENT $(sn,pn_1,pn_2,...,pn_n)$ SEGMENT DEFINITIONS

sn	Segment name to write on
pn_i	Subprogram or section name

SEGZERO$(sn,pn_1,pn_2,...,pn_n)$ FIRST SEGMENT DEFINITION

sn	Segment name to write on
pn_i	Subprogram or section name

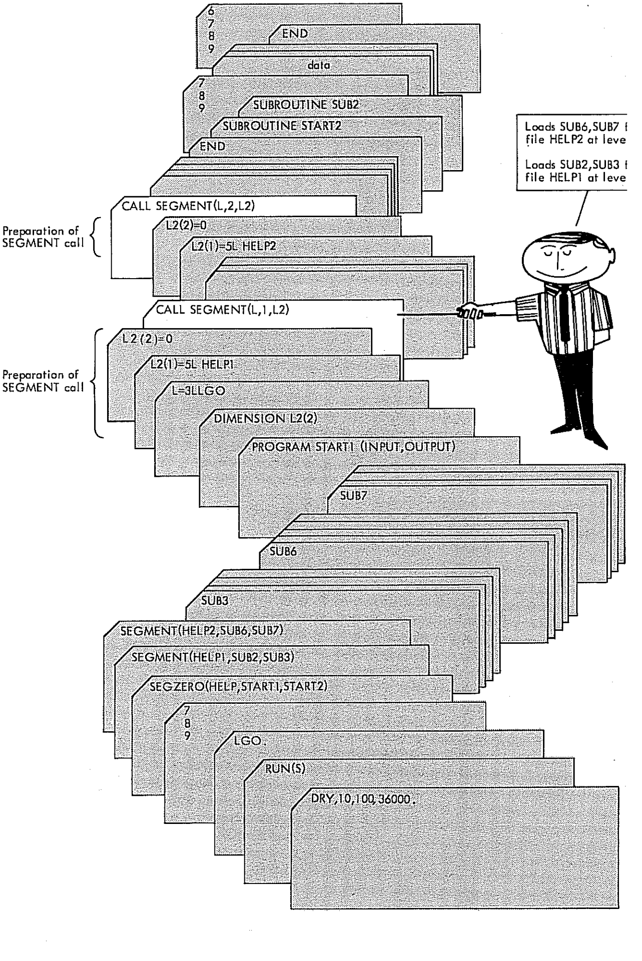
SECTION$(sn,pn_1,pn_2,...,pn_n)$ SECTION DEFINITION

sn	Section name to write on
pn_i	Subprogram name

OVERLAY$(fn,l_1,l_2,$cnnnnnn$)$ OVERLAY DEFINITION

fn	File name to write on; may be omitted after first, if same file
l_1	Primary level number; 0 for first overlay card
l_2	Secondary level number; 0 for first overlay card
cnnnnnn	Optional: load overlay $nnnnnn_8$ words from start of blank common

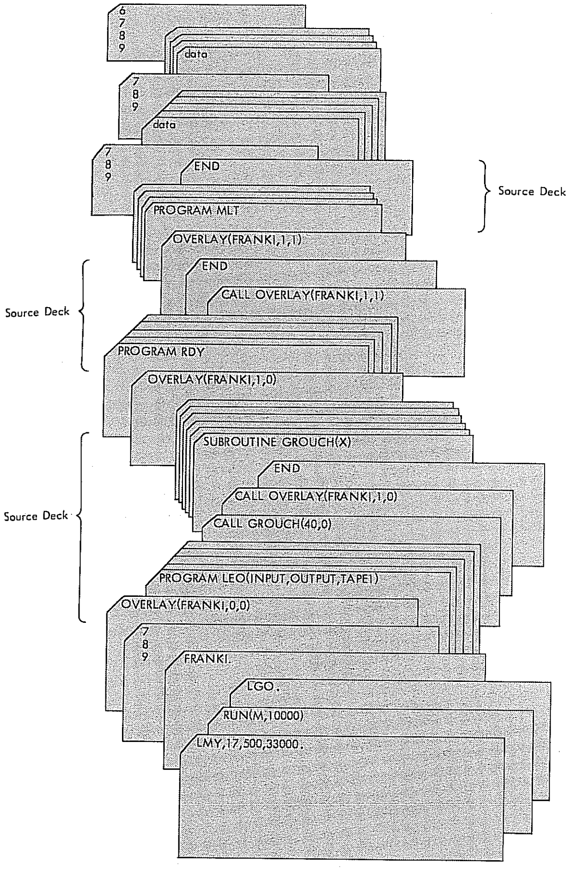
FORTRAN LOAD AND EXECUTE SEGMENTS



```
6
7
8
9
END
data
7
8
9
SUBROUTINE SUB2
SUBROUTINE START2
END
CALL SEGMENT(L,2,L2)
L2(2)=0
L2(1)=5L HELP2
CALL SEGMENT(L,1,L2)
L 2 (2)=0
L2(1)=5L HELP1
L=3LLGO
DIMENSION L2(2)
PROGRAM START1 (INPUT,OUTPUT)
SUB7
SUB6
SUB3
SEGMENT(HELP2,SUB6,SUB7)
SEGMENT(HELP1,SUB2,SUB3)
SEGZERO(HELP,START1,START2)
7
8
9
LGO.
RUN(S)
DRY,10,100,36000.
```

Preparation of SEGMENT call
Preparation of SEGMENT call
Loads SUB6,SUB7 f file HELP2 at leve
Loads SUB2,SUB3 f file HELP1 at leve

OVERLAY PREPARATION OF 0,0; 1,0; AND 1,1



NOTES

CONTROL DATA
CORPORATION

Documentation Department

3145 Porter Drive
Palo Alto, California 94304