NATIONAL INSTITUTE OF HYDROLOGY
ROORKEE

WORKSHOP

on

GROUND WATER MODELLING - TYSON-WEBER MODEL
(18 - 22 Nov.1985)

LECTURE	:	II
TOPIC	:	COMPUTER PROGRAMMING VAX-11/780
BY	:	1. SHRI S.K.Jain
		2. MRS. DEEPA KARAWADE
DATE AND TIME	:	19.11.85 11.45 A.M. - 1.00 P.M.
		19.11.85 2.30 P.M. - 4.15 P.M.

FORTRAN PROGRAMMING

1.0 INTRODUCTION

A computer programme is a set of instructions given to a computer to perform the desired computations. The languages which are used to write computer programmes are known as programming languages. Nowadays, a number of programming languages are available to a user. Some of them are: FORTRAN (FORmula TRANslation), COBOL (COmmon Business Orineted Language), BASIC (Beginners Allpurpose Symbolic Instruction Code), ALGOL (ALGOrithmic Language), PASCAL etc. In this course, thediscussion will be limited to FORTRAN language only. This language was developed by IBM corporation in 1956 and is the most popular and widely used programing language.

A computer programme written in a programming language is called a source programme. To run a source program on a computer, it is necessary to translate it into machine language so that it can be understood by the computer. This translation is done by another program called 'compiler'. The compiler checks whether the grammer of the language has been correctly followed or not. If not, it gives diagnostic error messages, otherwise it translates the source program into object programme.

Nowadays, a large number of versions of FORTRAN are available. The version which is being discussed here has been developed for VAX-11/780 machine and is called VAX-11 FORTRAN.

1.1 FORTRAN Character Set

The following set of character can be used in a FORTRAN programme: The letters A through Z and a through Z .

2. The numerals 0 through 9.
3. Special characters: (blank),
   =, + , -, , /, (,) , ., ^{11}, ^{1},
   , : : , <, >, %, & , comma, underline

Other printing characters can occur in a program as Holleuith characters.

2.0 FORTRAN Constants

Fortran constants are of following main types:

Integer constants and Real constants

Integer constants, also called fixed point constants, are whole numbers; they do not have any fractional parts. They may have either a + or - sign. By default, + sign is assumed. The value of an integer constant must be within the range -2147483648 to +2147483647.

Real constants, also called floating point numbers, are the numbers which have a decimal points.

A real constant can be written as
- as a real number
- as a real number followed by a decimal exponent
- as an integer constant followed by a decimal exponent.

A minus sign must appear before a negative real constant. A plus sign is assumed by default.

The real constants can be further classified in several sub-classes. A REAL* 4 constant occupies four bytes of VAX-11 storage. Typically its degree of precision is seven decimal digits. The magnitude of a REAL*4 constant cannot be greater than 1.7E38 and it cannot be less than 0.29E-38 approx. A REAL*8 constant is also called a double precision constant. On VAX-11, two types of REAL*8 constants are available: D-floating and G-floating. Both these occupy 8 bytes of VAX-11 storage. A D-floating number has a degree of precision of 16 decimal digits and it must lie within the range 0.29D-38 to 1.7D38 approx. A G-floating constant has a degree of precision of 15 digits and its magnitude must lie within the range 0.56D-308 to 0.9D308 approximately.

Other types of constants which can be put in this category are REAL*16, COMPLEX*8 etc.

A logical constant specifies a logical value which can be either true or false. Only following two logical constants are permissible:

.TRUE.

.FALSE.

3.0 VARIABLES

A variable is a symbolic name associated with a storage location. The value currently stored in that location is the value of that variable. Similar to the constants, variables are also classified into data type based on which, the storage requirements and the precision of the variables are determined.

In FORTRAN, by default, all the variable names beginning with I, J, K, L, M or N are assumed to be integer variables. Variables whose name begins with any other letter are assumed to be real variables. Thus AREA, VOLUME, TEMP are real variables while LENGTH, ITEM, NAME are integer variables. A variable name can have a maximum length of 31 characters but cannot begin with a numeral.

4.0 FORMAT OF A FORTRAN STATEMENT

A Fortran program can be fed to a computer typically either through punched cards or interactive terminals. Since the computer cards used earlier were 80 columns wide the convention of writing a program was designed to suit them. The same is followed now. In a Fortran statement, first five columns, are reserved for statement numbers. sixth column is used to indicate the continuation of the statement, columns 7 to 72 are used to write a statement and 73-80 columns are for comments. The letter 'C' or '*' in the first column indicates that statement to be comment and is ignored by compiler. The statements having character 'D' in the first column are compiled only when Debug qualifier is used. A statement number can be any integer number in the range 1 - 99999 . Decimal point and other letters are not permitted in the statement field. The statements need not be in sequence and no two statements can have same number.

If a statement can not be completed in 7-72 columns, it can be continued in the next line by putting a continuation mark in the 6th column of next card. Columns 73-80 are reserved for comments or sequence numbers of the program and are ignored by the compiler.

5.0 FORTRAN EXPRESSION

An expression consists of combination of single and multiple variables and/or constants with one or more operators. Operators specify the computations to be performed using the values of variables to yield a single value. Expressions can be classified as arithmatic, character and logical expressions.

Arithmatic expressions are formed with arithmatic elements arithmatic operators. Following five operators are used in Fortran

Operator	Function
+	Addition
-	Subtraction
/	Division
*	Multiplication
**	Exponentiation

The last three operators are called binary operators because they use two arithmatic elements. The first two operators can be used both as unary and binary operators. While evaluating an arithmatic expression, exponentiation operator gets first precedence, * and / get second precedence and + and - get the third. When two or more operators of equal precedence appear, they are evaluated in left-to-right order. The exception is exponentiation which is evaluated from right to left. Thus X * *   * * Z is evaluated as X * * (Y * * Z). The required order of evaluation is forced in a program by using paranthesis. The part of an expression which is enclosed in a parenthesis is evaluated first and then the resulting value is used for further computations.

5.1 Rational Expression

A rational expression consists of two arithmatic expressions or two character expressions separated by a rational operator which tests for a relationship between two expressions. The rational operators are

Operator	Meaning
.LT.	Less than
.LE.	Less than or equal to
.EQ.	Equal to
.NE.	Not equal to
.GT.	Greater than
.GE.	Greater than or equal to

Besides these, following logical expressions are used to perform logical operations

Operator	Use	Interpretation
.AND.	A. AND.B	The expression is true if and only if both A and B are true.
.OR.	A.OR.B	The expression is true if either A or B or both are true
.NOT.	.NOT.A	The expression is true if, and only if, A is false

6.0 ASSIGNMENT STATEMENT

The arithmatic assignment statement assigns the value of the expression on the right of the equal sign to the numeric variable or array element on the left of the equal sign. It has the form

v = e

where v is a numerical variable

e is arithmatic expression

The equal sign means ' is replaced by'. For example statement

I = I + 1

means that the current value of I is replaced by the previous value plus one. In a valid arithmatic statement, the entity on the left hand sign must always be an unsigned variable.

7.0 CONTROL STATEMENT

In a program, the statements are executed in the sequence in which they occur. This order can be changed by use of control statements. The control statements are discussed here briefly.

7.1 GO TO Statements

These statements transfer the control within a program unit. There are following three types of GO TO statements.

a) Unconditional GO TO statement for example GO TO 100

b) Computed GO TO statement for example
GO TO (10, 15, 50, 100), I

c) Assigned GO TO statement for example
GO TO 5, (150, 190)

7.2 IF Statements

The IF statements conditionally transfers the control or executes a statement or block of statements. The three type of IF statements are as follows:

7.2.1 Arithmatic IF This statement transfers control to one of three statements depending upon the value of arithmatic expression. It has the form.

```
If (e) S1, S2, S3
```

where e is an arithmatic expression. The control is transferred to statement number S1, S2 and S3 if the value of the expression is less than 0, equal to 0, or greater than zero respectively.

7.2.2 Logical IF statement

This statement conditionally executes a fortran statement in the following form.

IF(e) st

where e is a logical expression

st is a valid executable fortran statement except a DO statement, an END DO statement, an END statement and another IF statement.

7.2.3 Block IF statement

Block IF statement conditionally executes blocks of statements. The statements used to contruct block are IF THEN , ELSE IF THEN , ELSE and END IF. The END IF statement terminates the block IF construct.

7.3 DO statement

The DO statement is used to control the iterative processing where the numbers are iteratively executed a specified number of times. It has the form

DO St v = e1, e2 [,e3]

where st is the label of an executable statement which is the terminal statement of the DO loop.

v is an integer or real variable called the DO index or control variable. e1, e2, e3 are arithmatic expressions and are called initial, terminal and increment parameters respectively . If increment parameter is omitted, its default value one is taken, However, it cannot be zero.

The rules about the syntax and resting of DO loops valid for VAX-11 FORTRAN can be referred to in VAX-1 FORTRAN Language Reference Manual.

7.4 CALL statement

The CALL statement executes a subroutine or other external procedure. It can also specify an argument list for the subroutine. The example of a call statement is

```
CALL  ADD (x, Y, 1.0, MATR)
```

where ADD is the name of the subroutine to be called and X,Y, 1.0, MATR are the arguments.

7.5 RETURN statement

The RETURN statement transfers the control from a subprogram to the program which has called it.

7.5 PAUSE statement

This statement temporarily suspends the program execution, displays a message on the terminal and awaits for the reply. It has the form

```
PAUSE [disp]
```

where display is a character constant or a decimial digit string of one to five digits. By default, the message 'FORTRAN PAUSE' appears on the screen. The command CONTINUE' can be given to resume the execution, STOP to terminate the execution or DEBUG to do debugging.

7.6 STOP statement

The program execution is stopped upon encountering a STOP statement.

7.7 END statement

The END statement signifies the end of a program unit. It must be the last source line of a program unit.

8.0 Specification statements

The specification statements are non executable statements that are used to allocate and initialize variables and arrays etc. The important specification statements are briefly described here.

The DIMENSION statement defines the number of dimensions in a subscripted variable or array and the maximum number of elements in each dimension.

The IMPLICIT statement oversides the implicit data type of symbolic names.

The COMMON statement defines are or more contiguous areas of storage. This area can be stared by more than one program units in which the common declaration occurs. The DATA statement is used to assign initial values to variables, arrays, and array elements etc. before the execution of program.

The PARAMETER statement assigns a symbolic name to a constant.

Other statements available in this category .

include EQUIVALENCE, SAVE, EXTERNAL, INTRINSIC, PROGRAM-and BLOCK DATA.

9.0 SUBPROGRAMMES

These are programme units which can be involved from another programme unit. The subprogrammes can be of following three types:

a) Statement function subprogrammes,

b) Function subprogrammes

c) Subroutine subprogrammes

Besides these, a number of subprogrammes are also supplied with the language by the suppliers and can be called in a similar manner.

A statement function is computing procedure defined by a single statement. It has the form of an assignment statement . The statement function must appear in the same programme unit in which it has to be used. The following is an example of a valid statment function:

```
AREA (RAD)  =  3.14 * RAD  * * 2
```

A function subprogramme is a program unit consisting of a FUNCTION statement followed by a sries of statements. A function subprogramme returns a single value to the calling program and this value is assigned to the name of the function.

A subroutine subprogramme consists of a SUBROUTINE statement followed by a series of statements which define the computations to be performed.

A CALL statement is used to transfer control to the subroutine. Upon encountering a RETURN statement, the control is returned to the calling programme unit. When the control is transferred to the subroutine, the values of the actual argument are associated with the corresponding during arguments. ENTRY statements can be used to specify multiple entry points to a subroutine.

10.0 INPUT/OUTPUT statements

These statements are used to obtain input data from a device and to supply the output data.

The input statements are used to read input data from a fil or a terminal. Two statements, ACCEPT and READ are used for this purpose. The ACCEPT statement is used to read the data from an interactive terminal and the READ statement can be used to read the data from a file.

The statements TYPE, WRITE and PRINT are used for outputting the data. By default, TYPE statement displays the data on an interactive terminal, WRITE statement writes the data in a file and PRINT statement sends the data for printing on a printer.

FORMAT statements are associated with I/O statements and are used to specify the format in which the data transfer is to take place. Alternatively a format free I/O statement can be used in which the format of data depends on data type. The FORMAT statement is not being discussed here in further details. The details can be obtained from VAX-11 FORTRAN Language Reference Manual.

INTRODUCTION

The VAX series is DIGITAL's family of 32-bit mini-computer systems. The newest member of the family is VAX-11/780. Like its companion processor, the VAX-11/780 was designed to meet the needs of many users with large data bases and increased processing needs. VAX systems are high performance multi-programming computer systems which implement a 32-bit architecture, efficient paging memory management, and virtual memory operating system (VMS) to provide excellent applications performance and essentially unlimited program address space.

This system is ideally suited for a wide variety of applications including real time, batch, time sharing, commercial and program development.

The instruction set of processor (KA 780) includes floating point, fixed point decimal arithmetic and character string instructions. The software system supports programming languages that take advantage of these instructions to produce extremely efficient code.

The VAX/VMS virtual memory operating system provides a multiuser, multi-language programming environment on the VAX hardware. The floating point instructions and VAX-11 FORTRAN are ideal for real time and scientific computational environments. The processor executes variable length instructions in native mode and non privileged PDP-11 instructions in compatibility mode.

Symbol used and the function of some control keys of VAX-11/780 terminals:-

<RET> -	Press the RETURN key. It transmits the current line to the system for processing.
<DEL> -	Press the DELETE or RUBOUT key. It deletes the last character at the terminal screen.
<ESC> -	Press the ESCAPE or ALTMODE key.
$ -	DCL (Digital Command Language) prompt.
CTRL/C -	Hold down the CTRL key and press the character C. It is used mostly to interrupt the system during program execution.
CTRL/Y -	Hold down the CTRL key and press the character Y. It interrupts the command or program execution. It mostly functions as CTRL/C.
CTRL/Z -	Hold down the CTRL key and press the character Z. It terminates a file input from the terminal.
CTRL/U -	Hold down the CTRL key and press the character U. It deletes the current line.
CTRL/R -	Hold down the CTRL key and press the character R. It reprints the current line.
CTRL/I -	Hold down the CTRL key and press the character I. Its function is same as TAB key.
CTRL/S -	Hold down the CTRL key and press the character S. It suspends terminal output.
CTRL/Q -	Hold down the CTRL key and press the character Q. It restarts the terminal output that was suspended via CTRL/S.
-	Type a blank.

The NIH VAX-11/780 computer system configuration is as follows:-

[1] HARDWARE:

i) VAX-11/780 CPU with floating point accelerator.

ii) Two RM03 removable disc drives having 67 M byte formatted capacity.

iii) Two TE 16, 45 IPS, 800/1600 BPI tape drives.

iv) Two ADM-3A terminals.

v) One VT 100 terminal.

vi) LA-120 console subsystem.

vii) LA-120 Line printer.

viii) Tektronix 4027 Graphic terminal.

ix) Calcomp Model-31 Color graphics system.

x) VT-105 terminal.

xi) One RX-11 floppy drive.

[2] SOFTWARE:

i) VAX/VMS operating system version 3.2

ii) MACRO assembler.

iii) VAX-11 FORTRAN-77.

iv) PLOT-10 IGL.

v) SSP (Scientific Subroutine Packages)

vi) COBOL-74

vii) HEC Programs.

viii) BASIC-PLUS-TWO

ix) SORT/MERGE routines.

x) Runoff routines.

USING VAX-11/780:

To communicate with the VAX/VMS operating system a terminal connected to the computer is used in NIH Computer Centre. You tell the operating system what to do; by typing a command on terminals keyboard. The system responds by executing your command. If the system can not interpret what you type, it displays an error message at the terminal. When the command has been successfully executed, you type another.

LOGGING IN:

To begin a session at the terminal, you must log in. Logging in consists of getting the systems attention and identifying yourself as an authorised user. Before logging in to the system, the users must have the accounts which is given by the system manager. He will also give USER NAME and PASS WORD. The USER NAME identifies a user to the system.

The sequence of LOG IN is as follows:-

i) Switch on the terminal.

ii) Press <RET> or CTRL/Y.

iii) The system responds by prompting for the user name.

iv) Type your USER NAME and press <RET>.

v) The system prompts for user Password.

vi) Type your PASSWORD. It will not be echoed on the terminal.

vii) If the PASSWORD is correct, system prompts with WELCOME message.

viii) Now system is ready to accept and valid DCL command.

The log in sequence looks like this:-

```
<RET> or CTRL/Y
USER NAME      : DAVID    <RET>
PASSWORD       : DCOD     <RET>
WELCOME TO VAX/VMS VERSION 3.2.
$
```

ENTERING COMMANDS:

You can type the commands using upper or lower case letters or a combination of both e.g.

$ show time <RET> is a valid command. It displays the current date and time as follows -

```
1- MAY- 1985        11:50:49
```

OR

$ SHOW TIME <RET>

OR

$ Show Time <RET> serves the same purpose.

HELP COMMAND:

The HELP command assists the user by displaying command - specific information. The HELP command is a useful interactive reference tool for the user not having convenient access to reference manual e.g. to know about the PRINT command, use the HELP command as

$ HELP PRINT <RET>

Some information is displayed on the terminal which includes a synopsis of what the PRINT command does and the qualifiers of the command along with their default value.

LOGGING OUT:

After finishing the work, the user must give the LOG OUT command as follows:-

```
$ LOG OUT <RET>
```

and the user should wait until he receives the appropriate message.

Shutting off the terminal does not causes you to LOG OUT. If a user shuts a terminal without logging off properly, another user may be able to turn the terminal on later and use your account.

SOME DEFINITIONS:

1. FILE : -

A file is the basic unit of storage for VAX/VMS. All user information is stored in files, usually on auxiliary storage media such as tapes, disks etc. Actually a file is a collection of logically related data stored on disks or tapes. A complete file specification contains all the information the system needs to know and identify a file. A complete file specification has the format :

Device: [Directory Name] file name. File type: version

e.g.

```
DRAL: [DAVID] A.FOR: 1
```

2. DEVICE :-

It identifies the physical device on which a file is stored. Some device names are:

NAME	DEVICE
DRA	RM03 disk on controller A, unit 0
DRA1	RM03 disk on controller A, unit 1
MTA	TE16 magentic tape on controller A, unit 0.
MTA1	TE16 magnetic tape on controller A, unit 1.
TTA	Terminal on controller A, unit 0.
TTA5	Terminal on controller A, unit 1.

Note:- Everytime when a device name is specified as Colon(:) must follow it e.g. DRA0:, MTA1:, TTA5: etc.

3. FILE NAME:-

The file name is a set of one to nine alphanumeric characters.

File Type - A file type can be upto 3 alphanumeric characters and must be preceded by a period. However, the file type usually describes specifically the kind of data in the file and the system recognizes several default file types used for special purposes as follows:

File Type	Use
DAT	Data file
EXE	Executable program image (after linking)
FOR	Input source file for FORTRAN compiler
LIS	Output listing from a compiler
MAR	Input source file for MACRO compiler

File Type	Use
OBJ	Object module output from a compiler.
JOU	Journal file which is created if interruption occurs during the use of EDT editor.

4. VERSION NUMBER:-

The version number distinguishes a file from its other copies. It is connected to the file type by Semicolon (;) Whenever a user edits a file, its version number is automatically increased.

The system always takes the file with the highest version number as the default file. It can be overriden by explicitly specifying version number e.g. If in a directory the files are A. DAT; 1, A. DAT; 2, A. DAT; 3 and if

```
$ PRINT ␢ A. DAT  <RET>
```

command is given then the file A. DAT; 3 will be printed. Here to print a file A. DAT; 1, the command is

```
$ PRINT ␢ A. DAT; 1  <RET>
```

Use of Wild Card Character:-

A wild card character (*) can be used within a directory name, a file name, version no. or a file type. It can be used to match any number of characters including the null string e.g.

```
$ PRINT ␢ *. DAT; *  <RET>
```

will print all the 'DAT' files in the directory. Similarly

```
$ PRINT ␢ A. * ; *    <RET>
```

will print all the files in the directory whose file name field is A.

Abbriviating Commands:-

While typing the commands or qualifiers always there is no need to type the full words. In many cases only one or two characters act as full commands. Here it is necessary that the user must type atleast the minimum number of characters necessary to make the command unique. e.g. the SET, SEARCH and SHOW commands all begin with the letters 'S'. So to make these commands unique, you must type SET, SEA and SH respectively. Also RUN is the only command which starts from the character 'R', so to abbriviate RUN command can be R or RU.

Error Messages:

If a user enters a command incorrectly, the system displays an error message and prompts for a command line as if no command had been entered.

Creating and Running a Simple FORTRAN Program:

The method for creating and running a very simple FORTRAN Program (A) is as follows:-

i) Login as described earlier.

ii) Create the file A. FOR using the editor as follows:

```
$ EDIT ƀ A. FOR  <RET>
100  ACCEPT 10, ABS  <RET >
200  10 FORMAT (5A5)  <RET>
300  PRINT 20  <RET>
400  20 FORMAT ('' PROGRAM IS WORKING'')  <RET>
500  STOP  <RET>
600  END  <ESC>
*    E  <RET>
```

Now the file has been created. There may be some syntax errors in this Program. Such errors can be checked by compiling the program so give the command as

iii) $ FORTRAN ␢ A <RET>

If there is no error than after few seconds $ sign appears on the screen otherwise the errors are displayed on the screen. After successful completion of the program the object file (A.OBJ in this case) is created. The object file is necessary before a linker is used. The method of using the linker is as follows:

iv) $ LINK A <RET>

If the linking is done successfully then $ sign appears on the screen, otherwise the error is displayed on the screen. To run the program the command is

v) RUN ␢ A <RET>

After successful running of the program a $ sign appears on the screen. After which the results can be checked for the correctness. Now user can give another commands. This sequence can be explained using the flowchart shown here.

After running the executable image (A.EXE in this case) is created.

Now each time when the program is run there is no need of compiling and linking it again if A.EXE is in directory. Only the RUN command is required.

Debugging a FORTRAN Program:

The VAX/VMS operating system has a debugger with the help of which debugging can be done interactively. While using the debugger, the program must be compiled with /DEBUG and / NOOPTIMIZE qualifiers as follows:

```
$ FORTRAN/DEBUG/ NOOPTIMIZE ƀ A  <RET>
```

These qualifiers make the later use of the debugger program possible with this FORTRAN Program. When the compilation is complete use the /DEBUG qualifier to link the object module.

```
$ LINK/DEBUG ƀ A  <RET>
```

Now when the RUN command is used to execute the program image A.EXE, the debugger takes control and the debugging commands can be used to stop the execution of the program at a particular statement and examine and modify a variable.

IMPORTANT VAX/VMS COMMANDS

The VAX/VMS command language provides time sharing terminal users with an extensive set of commands for (a) interactive program development (b) device and data file manipulation (c) interactive and batch program execution and control.

General format of a command is

```
[$] Command - name [Qualifiers] [Parameter-1]...[Parameter-n]
```

Where Dollar sign [$] is must for all command procedure and it is not required for interactive mode because in this mode the system prompts $ sign. Brackets ([and]) are used to surround optionalvalues. The qualifier is used to modify the default action of a command. A plus sign (+) indicates concatanation of files or parameters. A hyphen (-) may be used for continuation on the next line.

For the convenience of the user, important commands are listed and described below:-

1. ALLOCATE

Format:ALLOCATE device - name [:] [Logical - name [:]]

The ALLOCATE command provides exclusive access to a device and optionally establishes a logical name for the device. Once a device has been allocated, other users can not access the device until the user specifically deallocates it or log out is done. e.g.

```
$ ALLOCATE DRA1:

      - DRA1 : ALLOCATED
```

The ALLOCATE command allocates a specific RM03 disk drive, unit 1 on controller A.

Note:- Don't try to allocate either users disk or system disk

2. APPEND

Format:

$ APPEND input-file-spec,.......output-file-spec.

The APPEND command adds the contents of one or more specified input files to the end of a specified output file. e.g.

```
$ APPEND  TEST.DAT  NEWTEST.DAT
```

The APPEND command appends the contents of the file TEST.DAT from the default disk and directory into the file NEWTEST.DAT.

```
$ APPEND/NEW/LOG *.TXT MEMO.SUM
```

The APPEND command appends all files with types of TXT to a file named MEMO.SUM. The LOG qualifier request a display of the specification of each input file appended.

3. ASSIGN

Format:

ASSIGN device-name [:] logical-name [:]

The ASSIGN command equates a logical name to a physical device name, to a complete file specification or to another logical name and places the equivalence name string in the process, group or system logical name table. e.g.

```
$ ASSIGN DRA1: [CHARLES] CHARLIE
$ TYPE  CHARLIE.DAT
```

The ASSIGN command associates the logical name CHARLIE with the directory name CHARLES on the disk DRA1. Subsequent references to the logical name CHARLIE results in the correspondence between the logical name CHARLIE and the disk and directory specified.

4. CONTINUE

Format:

CONTINUE

The CONTINUE command resumes execution of a DCL command, a program or a command procedure that was interrupted by pressing CTRL/Y or CTRL/C. The CONTINUE command can also serve as the target command of an IF or ON command in a command procedure or following a label that is the target of a GO TO command. e.g.

```
$ RUN  MYPRG
CTRL/Y
$ SHOW TIME
   1-MAY 1985    14:02:59
$ CONTINUE
```

5. COPY

Format:

COPY input-file-spec,.....output-file-spec.

The COPY command creates a new file from one or more existing files. The COPY command can: (i) COPY one file to another file (ii) concatenate more than one file into a single out put file (iii) COPY a group of files to another group of files. e.g.

```
$ COPY  TEST.DAT  NEWTEST.DAT
```

The COPY command copies the contents of the file TEST.DAT from default disk and directory into a file named NEW TEST.DAT.

```
$ COPY  *.COM [RAJ.DATA]
```

The COPY command copies the highest versions of files in the current default directory with a file type of COM to the subdirectory RAJ.DATA.

```
$ MOUNT  MTB1: TAPE
$ COPY  TAPE: *.*; *
```

The COPY command uses the logical name TAPE for the input file specifications, requesting that all files on the tape be copied to the current default disk and directory. All the files copied retain their file name and file types.

6. CREATE

Format:

```
CREATE  file-spec
```

The CREATE command creates a sequential disk file from records that follow the command in the input stream or create a directory file. e.g.

```
$ CREATE  A.DAT
Input line one ....
Input line two......
CTRL/Z
$
```

After the CREATE command is issued from the terminal, the system reads input lines into the sequential file A.DAT until CTRL/Z terminates the input.

```
$ CREATE SHAR.COM
$ DECK
$ FORTRAN  SHAR
$ LINK  SHAR
$ RUN  SHAR
$ EOD
$   SHAR
```

This batch job example illustrates using the CREATE command to create a command procedure from data in the input stream.

The DECK commands is required so that subsequent lines that begin with a dollar sign are not executed as commands, but are accepted as input records. Then the procedure is executed with the (Execute Procedure) command.

```
$ CREATE/DIRECTORY  DRA1: [SHAR]
```

The CREATE command creates a directory named SHAR on the device DRA1. Creating a directory requires privilege. Where as creating a subdirectory requires no privilege e.g.

```
$ CREATE/DIR   DRA1:[DASON.PROG].
```

7. DEALLOCATE

Format:

DEALLOCATE [device-name [:]]

The DEALLOCATE command returns a device that was reserved for private use to the pool of available devices in the system. e.g.

```
$ ALLOCATE MTB1:TAPE
      - MTB1 : ALLOCATED
$ DEALLOCATE TAPE
```

8. DEASSIGN

Format:

DEASSIGN [logical-name [:]]

The DEASSIGN command cancels logical assignments made with the ASSIGN, DEFINE, ALLOCATE or MOUNT commands.e.g.

```
$ ASSIGN  A.TMP Ø SYS $ OUTPUT
 :
 :
 :
$ DEASSIGN Ø SYS $ OUTPUT
```

9. DEBUG

Format:

DEBUG

The DEBUG command invokes the VAX-11 Symbolic Debugger after program execution is interrupted by CTRL/C or CTRL/Y. The program image being interrupted must contain the debugger, i.e. the image was linked with the /DEBUG qualifier and /or run with the /DEBUG qualifier e.g.

```
$ FORTRAN/DEBUG/NOOPTIMIZE SUBR
$ LINK/ DEBUG  SUBR
$ RUN  SUBR

        % DEBUG version 1.2 10 March 1985
% DEBUG-1- INITIAL, language is FORTRAN, scope and Module
set to 'SUBR'.
DBG > GO
ENTER  NAME :
ENTER  NAME :  uncontrolable loop
ENTER  NAME :
CTRL/Y
$ DEBUG
  DBG >
   .
   .
   .
   .
```

10. DECK

Format:

DECK

The DECK command marks the beginning of an input stream for a command or program. It is required in command procedures when the first non blank character in any data record in the input stream is a dollar sign ($). e.g.

```
$ FORTRAN  A
$ LINK  A
$ RUN  A
$ DECK
Input line one
Input line two
$ input line
    :
$ EOD
$ PRINT SUMMARY.DAT
```

11. DEFINE

Format:

DEFINE logical-name equivalence-name

The DEFINE command creates a logical table entry and assigns an equivalance name string to the specified logical name.

```
$ DEFINE PROCESS-NAME  LIBRA
$ RUN  WAKE
```

12. DELETE

Format:

DELETE/qualifier file-spec/queue-name/symbol-name

The DELETE command deletes one or more files from a mass storage disk volume.

Qualifier: /ENTRY with this qualifier deletes one or more entries from a printer or batch job queue.

/BEFORE The files created before the date specified in this qualifier are deleted.

/AFTER The files created after the date specifies in this qualifier are deleted.

$ DELETE *.COM; */BEFORE = 01-JUN/LOG

The command deletes all version of all files with types com that were either created or updated before June 1 this year.

13. DIFFERENCES

Format:

[illegible] -file-spec [compare-file-spec]

The DIFFERENCES command compares the contents of two disk files and creates a listing of the records that do not match.

$ DIFFERENCES/IGNORE = (COMMENTS SPACING) - COPY.COM

14. DIRECTORY

Format:

DIRECTORY [file-spec.....]

The DIRECTORY command provides a lists of files or information about a file or group of files.

$ DIRECTORY

$ DIRECTORY LOGIN.COM

15. DISMOUNT

Format:

DISMOUNT device-name [:]

The DISMOUNT command releases a volume previously accessed with a MOUNT command. Example.

```
$ MOUNT  MTAO: PAYVOL TAPE
$ DISMOUNT  TAPE
```

16. EDIT

Format:

```
$ EDIT/editor file-spec
```

The EDIT command invokes one of the VAX/VMS editor.

```
        editor = SOS
               = SLP
               = EDT
```

17. EOD

Format:

EOD

Signals the end of a data stream in interactive mode.

```
$ RUN PRG
  data
$ EOD
```

18. FORTRAN

Format:

FORTRAN file-spec.

The FORTRAN command invokes the VAX-11 FORTRAN compiler to complete one or more source program.

Example:

$ FORTRAN A+B/LIST, C+D/LIST = ALL/OBJECT = ALL

For the first compilation, the FORTRAN command the files A.FOR and B.FOR to produce an object module named A.OBJ and a listing file named B.LIS. For second compilation object module ALL.OBJ and a listing file named ALL.LIS are produced.

19. HELP

Format:

HELP [keyword [keyword]....]

The HELP command displays on the terminal information available in the system HELP files.

Example:

$ HELP ASSIGN

Information on ASSIGN command will display.

$ HELP ASSIGN PARAMETERS

Information on ASSIGN command parameters will display.

20. INITIALIZE

Format:

INITIALIZE device-name [:] volume-label

The INITIALIZE command formats and writes a label on a mass storage volume.

Example:

```
$ ALLOCATE MTB1:
     - MTB1 : ALLOCATED
$ INITIALIZE MTB1: SOURCE
$ MOUNT MTB1: SOURCE
  % MOUNT-I-MOUNTED,SOURCE mounted on - MTB1:
```

```
$ COPY *  FOR MTB1
$ DIRECTORY  MTB1:
      file detail will be displayed.
      .
      .
      .
$ DISMOUNT MTB1:
```

The volume (Tape or Disk) must be physically mounted before giving the command INITIALIZE and MOUNT.

21. JOB

Format:

```
$ JOB  User-name
```

The JOB command identifies the beginning of a batch job submitted through a system card reader.

Example:

```
$ JOB RAJ
$ PASSWORD RAJ
$ ON WARING THEN EXIT
$ FORTRAN SYS $ INPUT: AVERAGE
                  .
                  .
Fortran source Deck
                  .
                  .
$ LINK AVERAGE
$ RUN AVERAGE
data records for program average
                  .
                  .
$ PRINT AVERAGE
$ EOJ
```

22. LIBRARY

Format:

LIBRARY/qualifier library [file-spec,.....]. The LIBRARY command creates or modifies an object module library or a macro library or inserts, replaces or lists modulues, macros or global symbol names in a library.

Example:

$ LIBRARY/CREATE TESTLIB ERRMSG, STARTUP

The LIBRARY command creates an object module library named TESTLIB. OBJ and places the modules ERRMSG. OBJ and STRATUP. OBJ in the library.

Qualifier

/CREATE - create the object modules library

/INSERT - insert the modules in the library

/LIST - output written to a file specified.

23. LINK

Format:

The LINK/qualifier file spec.

The LINK command involves the VAX-11 linker to link one or more object modules into a program image and defines execution characteristics of the image.

Qualifier

/DEBUG - involes symbolic debuggen

/MAP/FULL - request full map of the image and MAP type of file created.

/RSX 11 - Involves the RSX-11 M task builder to build a RSX-11 M image

```
$ LINK/MAP/FULL DRACO, CYGNUS, LYRA
$ LINK/RSX11 AVERAGE
```

24. LOG OUT

Format:

LOG OUT

The LOG OUT command terminates an interactive terminal session.

Example:

```
$ LOG

        logged out at 10-March-1985 15:15:15.15
```

25. MACRO

Format:

MACRO/qualifier file-spec,....

The MACRO command involves the VAX-11 MACRO assembler to assemble one or more assembly language source programs. If qualifier/RSX 11 is specified, the MACRO command involes the MACRO-11 assembler, all other qualifier apply to both the VAX-11 and the MACRO-11 assembler.

Example:

```
$ MACRO ROUT
```

If this command is executed in a batch job, the assembler also creates a listing file named ROUT.LIS.

26. MAIL

Format:

MAIL file-spec.

Sends a message to another user or a group of users.

Example:

```
$ MAIL
MAIL > READ
      message
MAIL > SEND
TO : SYMGR
SUBJECT : SYSTEM TROUBLE
MESSAGE : 'TTC 5 (LA 120) IS NOT WORKING'
PROCESSING MAIL
NO ERRORS
- DONE -
```

27. MOUNT

Format:

```
MOUNT device-name,..... [volume-label,...] [logical-name[:]]
```

The MOUNT command makes as volume and the files or data it contains available for processing by system commands or user programs.

Example:

```
$ MOUNT MTA0: MATH06      '-TAPE
% MOUNT - I - MOUNTED  MATH 06 MOUNTED ON - MTA φ:
```

28. PASSWORD

Format:

```
PASSWORD password
```

The PASSWORD command specifies the password associated with the user name specified on a JOB card for a batch job submitted through the system card reader.

Example:

```
$ JOB SHASTRI
$ PASSWORD RAVI
      .
      .
      .
$ EOJ
```

29. PRINT

Format:

PRINT/qualifier file-spec.

The PRINT command queues one or more files for printing on a default system printer or on a specified device,

Qualifier/COPIES - No. of copies to be printed.

/HEADER - output block header at beginning of each file.

/DELETE - immediately remove the file from your directory and delete it after printing.

Example:

```
$ PRINT/COPIES=3/HEADER ABPHA.TXT/NO IDENTIFY.
```

30. PURGE

Format:

PURGE file-spec.

The PURGE command deletes all but not the highest numbered version or versions of a specified file or files.

Example:

```
$ PURGE *.COM
$ PURGE AVERAGE.FOR/KEEP=2
```

The PURGE command deletes all but the two highest numbered versions of the file AVERAGE.FOR.

31. RENAME

Format:

RENAME input-file-spec output-file-spec.

The RENAME command changes the directory name, file name, file type or file version of an existing disk file.

Example:

```
$ RENAME AVERAGE.OBJ GINGER.OBJ
```

32. RUN

Format:

RUN file-spec.

The RUN command places an image into execution in the process. The file type is assumed to be . EXE.

Example:

```
$ RUN LIBRA
```

33. SET

Format:

SET option/qualifier device-name.

where the options are

[NO] CONTROL-Y

DEFAULT

MAGTAPE

PROTECTION

QUEUE

TERMINAL

[NO] VERIFY

The qualifier depend upon the options and its characteristics the different qualifiers are

/SPEED	- speed of device.
/FOREIGN	- Mass storage volume as foreign
/DENSITY	- Density of Magnetic tape
/PRIORITY	- Priority of the process
/ENTRY	- Entry of the process in a particular queue
/WIDTH	- Width of the terminal
/PAGE	- Page length
/QUOTA	

Example:

i) $ SET [NO] CONTROL-Y

ii) $ SET DEFAULT DRA1:

iii) $ SET MAGTAPE MTB1:/DENSITY =1600

iv) $ SET PROTECTION =(GROUP=RWED,WORLD=R)

v) $ SET QUEUE SYS $ BATCH/ENTRY=211/HOLD/NAME=TEST.

vi) $ SET TERMINAL/WIDTH=132/SPEED=9600/PAGE=66 TTC2

vii) $ SET VERIFY

34. SHOW

Format:

SHOW option/qualifier

options

[DAY] TIME

DEFAULT

DEVICES

MAG TAPE

PRINTER

PROTECTION

TERMINAL

QUEUE

SYSTEM

The SHOW command displays information about the current status of the process, the system or devices in the system.

Examples:

```
i)        $ SHOW DAYTIME
              10-MARCH-1985
ii)       $ SHOW DEFAULT
              DRG1: [ALPHA]
          $ SET DRFAULT DRA ϕ:[RAJ.SYS]
          $ SHOW DEFAULT
             DRA ϕ :[RAJ.SYS]
iii)      $ SHOW MAGTAPE MTA ϕ:
          MTAϕ: UNKNOWN, DENSITY =8ϕϕ,FORMAT=NORMAL-11 ODD PARITY
iv)       $ SHOW PRINTER LPAϕ:
          LPAϕ: LP11, WIDTH=132, PAGE=64, NOCR,FF, LOWERCASE
          DEVICE spooled to DRAϕ:
v)        $ SHOW PROTECTION
          SYSTEM=RWED,OWNER=RWED,GROUP=RE, WORKD=NO ACCESS.
vi)       $ SHOW QUEUE/DEVICES
          *DEVICE QUEUE " LPAϕ:" FORMS =O GENPRT GFLAG
          *DEVICE QUEUE " LPBϕ:" FORMS=ϕ GEN PRT GLAG
```

35. SORT

Format:

SORT/qualifier input-file-spec. output-file-spec.

The SORT command invokes the SORT utility program to records in a file into a defined sequence and to create a new file of the recordered records.

```
$ SORT/RSX11 CUSTOMER.FIL/FORMAT=(FIXED,80) ALPHA,SRT/KEY=(1.20)
```

The SØRT command requests a default alpha numeric sort on the records in the file CUSTOMER. FIL. The SORT program sorts the records based on the contents of the first 20 character in each record and writes the sorted list into the output file ALPHA.SRT.

36. SUBMIT

Format:

```
SUBMIT file-spec.....
```

The SUBMIT command enters a command procedure in the batch job queue.

Example:

```
i)         $ SUBMIT AVERAGE
           Job 112, entered on queue SYS $ BATCH.
ii)        $ SUBMIT/NAME = BA24/HOLD TEST ALL
           Job 467 entered on queue SYS $ BATCH
```

The SUBMIT command enters the procedure TEST ALL.COM for processing as a batch job but in a HOLD status. The Job will not be released until the SET QUEUE/RELEASE command is issued. The/NAME parameter requets that the batch job be identified as BATCH:24.

37. TYPE

Format:

```
TYPE file-spec.,....
```

The TYPE command displays the contents of a file or group of files on the current output device.

Example:

```
$ TYPE COMMON.DAT
```

38. UNLOCK

Format:

UNLOCK file-spec......

The UNLOCK command makes accessible a file that became in accessible as a result of being improperly closed.

Example:

```
$ TYPE TEST FILE.OUT
% TYPE-E-OPEN IN, error opening DRA1:[MALCOLM] TEST FILE
   OUT, 3 as input
- SYSTEM-W-FILEZ LOCKED, file is deaccessed locked.
$ UNLOCK TESTFILE 1.OUT
$ TYPE TESTFILE .OUT
```