AND

Alpha-Numeric Directory

AND . . . The AlphaNumeric Director System

The AND System provides the facility for storing and updating symbolic records on the disk and magnetic tapes attached to the Carnegie Tech G-21 computor. AND is useful for large programs originating from punched cards, and for nearly all programs originating from remote teletype stations. The AND system is both a general librarian for storing files of alphanumeric text and a text editor for altering that text.

I. Introduction

AND stores and modifies alphanumeric information which we will call "text" in groups of 80 characters. Each 80 character group corresponds to the contents of a standard 80-column card and will be called a "card image". AND treats each card image of text as a unit, never examining its contents (except the language field, columns 1 and 2). Any alphanumeric information can be treated as text by AND: for example, GATE, ALGOL, THAT, or IPL programs or data.

AND stores text in the G-20's disc memory in logical groups called "records". Each AND record is filed under the usage number of the programmer who created it. There may be many different AND records filed under any usage number, so each is distinguished by a "file number", assigned by the programmer when he first creates the record. File numbers may be used in any order and may be integers between 0 and 127, for each usage number.

AND maintains a master Directory which is used to locate current AND records. The Directory entry for each AND record contains the number pair: (Usage Number, File Number) which distinguishes the record, as well as the length and location of that record on disc (or on tape). The programmer selects a particular record by using the AND instructions "USER" and "FILE".

As an example of the use of AND, assume that a programmer needs to debug a large ALGOL program and does not wish to run the entire program deck through the card reader many times. His program and possibly some test data must first be stored on disc as an AND record. For this purpose he will prepare a card deck including: (1) text of his program and (2) suitable AND instruction cards to store this text into a new record under his usage number and a file number of his choice. When the programmer subsequently finds errors in his program, he can use AND to alter or delete any card images and insert any new text he desires; the card deck for these changes will contain only the AND instruction cards and the new text.

Any AND program can end with a RUN instruction; for example, the instruction RUN , ALGOL, TAPE" causes the ALGOL translator to be loaded and executed, taking its card images from the AND information on the disc rather than from the card reader.

AND is essential for running a program from a remote teletype, since AND saves typing the entire program more than once. AND operations are performed from a remote teletype just as from a card deck, since the G-20 treats each line at a teletype as simply a card image. Although this write-up will always refer to AND instructions and text as if they originiated from a card deck, it should be understood that each "card" could be a card image typed at a remote teletype, or the result of a previous AND run.

AND is called into operation by a job card which contains AND as the system name, followed by cards containing AND instructions and cards containing text. No card can contain both text and AND instructions, since text is handled without ever looking at its contents.

A programmer may use AND to take card images from any existing AND record; however, AND will allow a programmer to create new records only under his own man number, and will not let him change in any way the contents of any other programmer's AND records.

II. The Dump-Count

AND uses a temporary work area on disc called the AND "Scratch Area". Under control of the instruction cards, AND collates into the Scratch Area any desired combination of card images from existing AND records with card images of new text. The text thus collated can be copied from the Scratch Area back onto AND records on disc by a DUMP instruction; the "dumped" text may replace an existing record or may be assigned a new program number and become a new AND record. A RUN instruction may cause any system to be loaded and executed as if called from a job card, but with a monitor switch (|15)set to take input either from the AND Scratch Area or directly from the AND record.

There is a counter, called the "dump counter", associated with every AND record. Each time a DUMP instruction writes information onto an AND record, the associated dump counter is incremented by one. The value of this counter, called the "dump count", is designed to protect users against possible results of disc failures.

To access any AND record, the user must specify the current value of the dump count as well as the usage number and file number. If he gives the wrong dump count, AND will print the correct dump count and subsequently give an error when a DUMP is attempted. The dump-count will never exceed 31 and will cycle between 1 and 31. A program will have zero dump-count only when first created; thus, if a program with dump-count 31 is DUMPed on, its dump-count will become 1.

We will explain the need for the dump count by an example. Let us suppose that on Monday morning (as every morning) we copy the AND record area of the disc onto an emergency backup tape. Monday afternoon, the user makes extensive changes in his file and dumps it back

onto the same record. Monday night there is a serious disc failure which results in losing all the information from the disc. Our recovery procedure will be to copy onto the disc the contents of the last backup tape--the one created Monday morning. Suppose that Tuesday afternoon the man again uses AND to modify his program. His AND deck will contain serial numbers referring to his Monday afternoon DUMP; if this deck is applied to the file which is now on the disk (the Monday morning version), chaos will result.

The dump count provides an automatic mechanism for preventing this chaos. Since the user has done a DUMP onto the record after the backup tape was created, his Dump Count will not agree with the dump count attached to the record now on the disc, and his program will be faulted on Tuesday afternoon. If, on the other hand, the record has not been changed since Monday, then the dump count will not have changed and the disc failure will be irrelevant to the user.

III. AND Editing Operations

AND attaches four-digit sequential line numbers, or "serial numbers", to the card images which it collates into the Scratch Area. When (or if) these card images are subsequently copied back into an AND record, the new serial numbers are copied also. Every card image in AND records and in the Scratch Area has an associated serial number; the first image in a record always has serial number 1. Whenever the card images are printed, either by AND or by another system which takes its input from AND, a serial number is printed to the right of each card image as if it were in colums 81 to 84 of the card.

The serial numbering for AND card images will give unique serial numbers up to card 41999. This system is as follows:

CARD NUMBER	WILL PRINT AS
0... 9999	0000...9999
10000...10999	$_{10}$000...$_{10}$999
11000...11999	A000...A999
. .	. .
. .	. .
. .	. .
36000...36999	Z000...Z999

CARD NUMBER	WILL PRINT AS
37000...37999	1000...1999
38000...38999	←000..←999
39000...39999	→000..→999
40000...49999	¬000..¬999
41000...41999	,000...,999
- - - - - - - - - - - - - - - - - - - -	
42000...42999	0000...9999
Etc.	Etc.

To explain the operation of the various AND editing instructions, we will use two pointers, δ and σ.

δ is the record pointer, whose value is the serial number of the card image which is to be moved next into the scratch area. δ refers to the AND record most recently selected by USER and FILE instructions.

σ is the scratch area pointer, whose value is the new serial number which will be assigned to the next card image entered into the scratch area. As each card image is entered into the scratch area, σ is always increased by 1.

The basic text manipulation operation of AND simply moves card image number δ from the current AND record into the Scratch Area to become image σ, attaches the new serial number σ to the image in Scratch, and finally steps both δ and σ by 1. The AND collation instructions iterate this operation to move blocks of images into the Scratch Area. The collation instructions also generally reset δ to delete, alter, and rearrange images as desired.

Each card image of new text is simply moved into the Scratch Area, as the next image and assigned the new serial number σ; then σ is stepped by 1.

The pointer σ can be reset to 1, "clearing" the Scratch Area, so that multiple collations can be performed during a single AND run.

AND can be instructed ("PRINT ALL") to print each card image as it is entered into the Scratch Area; in this case, the new serial number σ is printed beside each image.

IV. Detailed Explanation of AND Instructions

A. General Form of AND Instruction Cards.

1. Columns 1, 2 (the "language field") may contain "AN" or be blank.
2. Columns 3-80 may contain any number of instructions, separated by semicolons. Blank columns are completely ignored.
3. Each instruction must be entirely on a single card.
4. Comment convention: AND ignores everything to the right of a | ("bar") on a card. Thus every instruction will be terminated by a semicolon, a |, or the end of the card. Since the end of the card is a delimiter, the last instruction on a card need not be followed by a semicolon.

B. Basic AND Control Instructions

Before any instruction which refers to an AND record is given, the particular record must be selected with the pair of instructions 'USER', 'FILE'

(1) ...USER <Usage Number>

<Usage Number> is punched just as on a job card in the form <letter> <letter or digit> <2 digits> <man number> where <man number> ::= <2 letters> <2 digits>. Set <usage number> of next AND record to be selected, and <File Number> undefined. Every AND run begins with an implied USER Instruction:
USER <Usage Number on job card>
Thus, if you are using only your own records, you will not need any USER instructions in your program.
Example:

```
$$      2       10          AND       PHOlJPOl

AN FILE         62/1;       RUN, ALGOL, TAPE;
```

(2) ...FILE <File Number> / <dump-count>

(a) <File Number> must be an integer between 0 and 127, inclusive.

<dump-count> must either be an integer equal

to the current dump-count of the record, or else be the word: PERIL

(b) Using the <usage number> specified by the most recent USER instruction, FILE searches the AND DIRECTORY for a record filed under

(<Usage Number> , <File Number>).

If such a record does not exist, AND creates a Directory entry for it if the man number part of <Usage Number> agrees with that on the Job card.

(c) FILE sets $\delta \leftarrow 1$.

(d) When selecting a different File Number with the same Usage Number, it is not necessary to repeat the USER instruction before the new FILE instruction.

(e) Each time a DUMP instruction is executed, the dump-count associated with the AND record being dumped into will be increased by 1. A newly-created (and therefore empty) record has dump-count 0. To access a record, the user will generally need to supply the current dump-count of the record in the FILE instruction.

Examples: FILE 2/3;
FILE 2/PERIL;

The user can access AND records "PERILously" without supplying the correct dump-count; however, AND will not permit him to DUMP onto any AND record unless he has supplied correct numerical dump-counts in all FILE instructions in the run.

AND will always print the correct dump-count whenever a DUMP is executed and whenever the user does not supply the correct dump-count on a FILE instruction. The correct dump-count will be printed in these circumstances regardless of the PRINT option. If the AND program originated at a remote station, the dump count will be typed out there even if no TTYPE instruction has been given.

A dump-count of the form: PERIL will allow a PURGE to be

performed. Normally the record to be purged will be obsolete because it has been modified and dumped onto another, "derived" record. It is recommended that prior to the PURGE instruction the user execute a FILE instruction to select the derived record and check its dump count.

(3) ... DUMP (No parameters)

DUMP <File Number> / <integer>

(a) DUMP writes the (σ -1) card images which are now in the AND Scratch Area onto the AND record named by the most recent (USER, FILE) instruction pair. AND will fault the DUMP instruction if the Man Number of the record to be written is not the same as the Man Number on the job card. In other words, a programmer is permitted to dump ONLY onto his own records.

(b) At the completion of the DUMP operation:
(x) The pointer σ is unchanged, so that more card images can be collated onto the end of the Scratch Area:
(y) The pointer δ has been reset to 1 and the new record which has just been written is selected.

(c) The DUMP writes an "End-of-File" image immediately after the last card image in the AND record. This will act as an End-of-File signal if a system which is taking its input from this AND record subsequently attempts to read beyond the end of legitimate information.

(d) AND has a feature to help save the programmer from his own catastrophic errors, such as altering the second image in a long record and dumping back onto it without doing a GET TO $ instruction.

The user can create a directory entry for

file number 0 (zero) under his usage number, by executing the instruction·

FILE 0/0 ;

Whenever he subsequently executes a DUMP onto any of his other records (with file numbers 1, ... 127), the original copy of the record will be S*A*V*E*D and will automatically appear in his directory under file number 0. At the same time, the message:

SAVED ON FILE 0

will be printed. This action will not change the dump-count of file 0.

File 0 will behave otherwise as a normal AND record. Thus, it can be selected, edited, and explicitly dumped upon in the normal way. Explicit dumps on FILE 0 will increment its dump-count as if it were any other record, and of course in this case the original copy can't be saved on FILE 0. If FILE 0 is never explicitly dumped upon, then it will always have dump-count = 0.

If the programmer has no need or desire to use the automatic save on FILE 0, he should simply not create a directory entry with file number 0, or purge the one he already has.

If there is sufficient available space to dump a record but not sufficient to save the old record on FILE 0, AND will not save it, and will inform the programmer of this with the message:

NO SAVE ON FILE 0

It should be emphasized that while FILE 0 will behave under normal AND operations like records with numbers 1, ... 127, FILE 0 should be regarded as volatile storage which has a lifetime of only a few days. As part of the regular

policing of AND storage which the Computation Center has instituted, records under FILE 0 which have not been altered for several days (a minimum of 48 hours) will automatically be returned to AND available space. This will be done without changing the dump-count of FILE 0.

(e) DUMP may have an optional parameter. See Section G

(4) ... NO SAVE (no parameters)

The instruction NOSAVE will keep the contents of FILE 0 safe over the next DUMP; that is, there will be no attempt to save on FILE 0, the previous contents of the file being DUMPed on.

(5) ... RUN , <System Name>
RUN , <System Name> , Card
RUN , <System Name> , Tape
RUN , <Logical File Type>
RUN , <Logical File Type> , Card
RUN , <Logical File Type> , Tape
RUN , <System Name> , <Logical File Type>
RUN , <Logical File Type> , <Logical File Type>

(a) Writes an AND End-of-File image after the last image in the Scratch Area.

(b) Terminates the AND run by loading and executing the system specified by the first parameter.

Specifying a Logical File Type for the first parameter will cause AND to load and execute the system corresponding to the Logical File Type. (Sometime in the future, it will be possible for a user who writes his own system to have it put on an AND record. When he wishes to use the system as the <System Name> parameter to AND's RUN instruction, AND will have the monitor execute the system as if it were a job card callable system, e.g., ALGOL, GATE, IPLV. The current monitor's system load routine does not have the ability to load a system from a Logical File Type. When monitor is modified to do this, an announcement will be made. Currently, any attempt to

use a Logical File Type as the first parameter to the RUN instruction will cause an AND error).

The time and paper limits on the original job card will be the total limits over both the AND run and the system run which follows.

(c) The second parameter controls the setting of the "Card-Read Switch" |15 in the Monitor, as follows:

<empty>: No change in previous setting of |15 (i.e. the system to be run will take its images from the same source from which AND has been taking its instructions).

,CARD: Set |15← 0(take input from cards).(The Monitor tape parameters will be set to point to the currently selected AND record, so the running system can take input from cards and then subsequently change |15 to take input from this AND record).

, TAPE: If $\sigma \neq 1$ then |15 ←-1 (i.e., if preceding AND run resulted in collating one or more card images into AND Scratch Area, then set |15 to take input from Scratch Area), else |15 ←+1, and set the Monitor tape parameters for current AND record; (i.e., if preceding AND run included only USER, FILE (and possibly PRINT) instructions, set Monitor tape parameters to take input directly from the selected AND record). Note that "RESET 1" will set $\sigma \leftarrow 1$ and will force "RUN, <system>, TAPE" to take input directly from AND record.

,<Logical File Type>: Set switches to take input from the beginning of the specified Logical File Type.

C. Collation Instructions.

The following instructions collate card images of text from

the card reader. All instructions except TEXT require that a (USER, FILE) pair have been executed previously to select a record.

In describing the form and function of the AND instructions, we will denote the numerical parameter values by letters n, p, q, and r, and their form as punched into a card by [n], [p], [q], and [r]. Thus, if the form of an instruction is 'GET [n]', then the value of the parameter is n.

Finally, M will stand for the total number of card images, or the maximum serial number, in the selected AND record. The End-of-File image which AND automatically places at the end of each record would be image (M+1); however, it cannot be collated by any AND collation instruction.

The basic collation instructions will now be described.

(6) ... SET [n]

(a) Sets AND record pointer:

$$\delta \leftarrow n$$

(b) Error unless: $1 \leqq \delta \leqq M+1$ where

M = maximum Serial Number in record.

(7) ... GET [n]

(a) Fetch n consecutive card images from the selected AND record starting with current Serial Number δ ; write them into the Scratch Area as the next n images with Serial Numbers $\sigma, \sigma + 1, \ldots, \sigma + n - 1$.

(b) The final effect of GET on the pointers is:

$$\delta \leftarrow \delta + n \; ; \; \sigma \leftarrow \sigma + n$$

(8) ... GET TO [q]

Fetch into the Scratch Area all images from the current value of δ up to and including image number q; i.e.,

GET TO [q] = GET [q - δ + 1]

The final effect of GET TO [q] on the pointer is:

$$\sigma \leftarrow \sigma + (q - \delta + 1) \; ; \; \delta \leftarrow q + 1;$$

"GET TO $" is also allowed; it will fetch

(M - δ + 1) images, from the current value of
to the end of the record, inclusive. Thus, the
single instruction GET TO $ will fetch all
the rest of an AND record into the Scratch Area.

(9) ... TEXT (no parameter)

(a) TEXT reads the following cards as input text, entering images into the Scratch Area and assigning consecutiye Serial Numbers to it.

Only AN in language field will terminate text; and, the card which terminates the text will be scanned for AND instructions.

The instructions SET, GET, GET TO, and TEXT just described form a complete set of basic text manipulating operations in AND; these instructions are, however, generally too elementary for convenience in actual editing programs. Therefore, AND contains a set of collation macro-operations: LOAD, INSERT AFTER, DELETE, ALTER, AND PUT, each of which is simply a particular combination of the basic operations. The last four macros are particularly convenient since they each per-

form one of the basic editing functions of : inserting new text, and deleting, altering, and rearranging old text; in addition, these forms are independent of other editing macro operations which precede and follow them. Thus, if only these four operations are used, it is usually possible to change the AND program to include new corrections by simply inserting the necessary collation instructions into the AND deck, without changing any existing AND cards. These four all start with a GET TO operation whose parameter must be non-negative; this means that these operations must be executed in order of increasing record serial numbers δ.

The word TO is used as a parameter delimiter in most of the macro collation instructions, as for example:

DELETE 64 TO 67 ;

TO is always used in the inclusive sense, so that this instruction deletes 64,65, 66, and 67.

The symbol $ can be used as one of the parameters to any of the macro instructions and stands for M, the serial number of the last card image in the record. That is, $ is [M]. AND looks up M, the number of card images in the record, in the Directory and uses its value for $.

The parameters for which $ can be used are denoted by q and r in the following.

(10) ... LOAD [p] TO [q] ≡ SET [p] ; GET TO [q]
LOAD [q] ≡ LOAD [q] TO [q]
In either case, [q] may be $; for example,
LOAD 63 TO $ loads all card images from 63 to the end of the record, inclusive.
Parameters to LOAD are listable in the following form:
LOAD [p] TO [q], [r] TO [s]; ≡ LOAD [p] TO [q];
LOAD [r] TO [s];

(M - δ + 1) images, from the current value of
to the end of the record, inclusive. Thus, the
single instruction GET TO $ will fetch all
the rest of an AND record into the Scratch Area.

(9) ... TEXT (no parameter)

(a) TEXT reads the following cards as input text, entering images into the Scratch Area and assigning consecutive Serial Numbers to it.

Only AN in language field will terminate text; and, the card which terminates the text will be scanned for AND instructions.

The instructions SET, GET, GET TO, and TEXT just described form a complete set of basic text manipulating operations in AND; these instructions are, however, generally too elementary for convenience in actual editing programs. Therefore, AND contains a set of collation macro-operations: LOAD, INSERT AFTER, DELETE, ALTER, AND PUT, each of which is simply a particular combination of the basic operations. The last four macros are particularly convenient since they each per-

form one of the basic editing functions of : inserting new text, and deleting, altering, and rearranging old text; in addition, these forms are independent of other editing macro operations which precede and follow them. Thus, if only these four operations are used, it is usually possible to change the AND program to include new corrections by simply inserting the necessary collation instructions into the AND deck, without changing any existing AND cards. These four all start with a GET TO operation whose parameter must be non-negative; this means that these operations must be executed in order of increasing record serial numbers δ.

The word TO is used as a parameter delimiter in most of the macro collation instructions, as for example:

DELETE 64 TO 67 ;

TO is always used in the inclusive sense, so that this instruction deletes 64,65, 66, and 67.

The symbol $ can be used as one of the parameters to any of the macro instructions and stands for M, the serial number of the last card image in the record. That is, $ is [M]. AND looks up M, the number of card images in the record, in the Directory and uses its value for $.

The parameters for which $ can be used are denoted by q and r in the following.

(10) ... LOAD [p] TO [q] ≡ SET [p] ; GET TO [q]
LOAD [q] ≡ LOAD [q] TO [q]
In either case, [q] may be $; for example,
LOAD 63 TO $ loads all card images from 63 to the end of the record, inclusive.
Parameters to LOAD are listable in the following form:
LOAD [p] TO [q], [r] TO [s]; ≡ LOAD [p] TO [q];
LOAD [r] TO [s];

(11) ... INSERT AFTER [q] ≡ GET TO [q] ; TEXT

Here [q] may be $.

(12) ... DELETE [p] TO [q] ≡ GET TO [p-1] ; SET [q + 1]

DELETE [q] ≡ DELETE [q] TO [q]

In either case, [q] may be $.

Parameters to DELETE are listable in the same form as those for LOAD.

(13) ... ALTER [p] TO [q] ≡ DELETE [p] TO [q] ; TEXT

ALTER [q] ≡ DELETE [q] ; TEXT

That is, ALTER means a DELETE followed by a TEXT. The new text which follows can contain any number of images, so that ALTER need not make a 1-to-1 replacement of the deleted images. [q] may be $.

(14) ... PUT [p] TO [q] AFTER [r] ≡

GET TO [r] ; LOAD [p] TO [q] ; SET [r + 1]

PUT [q] AFTER [r] ≡ PUT [q] TO [q] AFTER [r]

PUT [p] TO [q] ≡ PUT [p] to [q] AFTER [q-1]

Here either [q] or [r] can be $.

Parameters to PUT are listable in the same form as those for LOAD.

Example: to move images 64, ..., 67 to a new place after image 119, use:

```
DELETE 64 TO 67 ;
PUT 64 TO 67 AFTER 119 ;
which is equivalent to:
GET TO 63 ; SET 68 ;
GET TO 119; SET 64 ; GET TO 67 ;
    SET 120
```

Notice that PUT itself simply makes a copy of the images without deleting them from their original place.

D. Print Control Instructions

(15) ... PRINT <Print Mode>

The PRINT Instruction may have any of the following mnemonic parameters:

	print mode	Meaning
Normal	TEXT	Print all card images which are read from cards by a TEXT instruction, as well as all cards containing AND instructions.
	ALL	Print all card images, both existing images from records and new text from cards, as they are entered into the Scratch Area. The AND instruction cards are printed as well.
	<Empty>	Print only AND instructions.
	NO	Suppress all printing.
	SCRATCH	List on the printer the entire current Scratch Area contents, leaving the pointers and the regular print options unchanged.

Note that AND instruction cards will always be printed unles PRINT NO turns off all printing. If no PRINT instruction is given, AND assumes: PRINT TEXT.

(16) ... PAGE (no parameters)

Upspaces printer to new page.

(17) ... TTYPE or TTY (no parameters)

Set the Monitor to deliver all images being printed by the AND run (as determined by the PRINT instruction(s) on the remote station from which the current AND run originated.

TTYPE will be ignored if the run did not come from a remote teletype.

TTYPE [k] or TTY [k]

Same as TTYPE, except types out only the first k columns of each printed card image on the remote teletypes; here $0 \leq k \leq 66$. The AND serial will be typed out on the left of each card image, but will continue to appear on the right side of the on-line listing.

(18) ... NO TTYPE (no parameters)

Set the Monitor to deliver AND output images only to the printer, not to the remote teletype. At the beginning of every AND run, the Monitor will be in the NO TTYPE state.

(19) ... PEEK <integer>

PEEK <empty>

The AND instruction

PEEK <integer> or PEEK <empty> (where <empty> implies 47 (1 page full))

gives the user a simple way of reviewing all his AND records so that he can purge (or put into COLD STORAGE) those which are no longer necessary.

PEEK will print out the first <integer> lines of each AND record filed under his usage number. More exactly, it will print <integer> or <NUMBER OF IMAGES IN RECORD>, whichever is smaller. Each program will be listed starting on a new page.

PEEK will be most useful if you make a habit of placing comment card images at the beginning of each AND record, telling when the record was created, its parentage, and why it was created (if you don't know why, then perhaps you should not be using AND record space...). The form of these comments is determined by the comment convention of the language whose text is stored in the record, of course.

PEEK will not disturb the scratch area or the scratch pointer. Following a PEEK, however, a new program instruction must be given to access an AND record.

You will be allowed to PEEK only at your own records. If you execute a USER instruction which selects a usage number different

from that on the job card, and then try to execute PEEK, you will get an AND error message:

YOU ARE A PEEK-ING TOM.

E. Logical File Table Instructions

(20) ... FILE <file number> / <dump-count> (<integer>)

It is now possible to make entries in the monitor Logical File Table from AND. If the FILE instruction or the FILE <File Number> / <dump-count> instruction is followed by (<integer>), the currently selected (USER, FILE) pair will be defined as Logical File Type <integer>. If the Man Number of the (USER, FILE) pair defined as a Logical File Type does not agree with the Man Number on the job card, a "read only" flag will be set for that Logical File Type. Thus, if file 17/3 of user MM01PS01 is desired to be Logical File Types 18 and 14, the following AND instructions should be executed:

```
USER MM01PS01 ;
FILE 17/3 (18) ;
FILE (14) ;
```

If an AND record has been set up as a Logical File Type and this record is DUMPed on or PURGEd during an AND run, the Logical File Table will be altered correspondingly. If a DUMP is executed on a record appearing in the Logical File Table, the Logical File Type (s) corresponding to that record will be updated, i.e., if the AND block number and/or the length of the record is changed, the Logical File Table will be altered. If a PURGE is executed on a record appearing in the Logical File Table, the Logical File Type(s) corresponding to that record will be set undefined.

The Logical File Type must be less than 20 and greater than 1. When a RUN instruction is executed, Logical File Type 1 is automatically set to the first unused block of Scratch, that is, the first block following the program or data in Scratch.

(21) ... CREATE <integer> BLOCKS

CREATE <integer> CARDS

CREATE <integer> BLOCKS FOR <file number> / <dump-count>

CREATE <integer> CARDS FOR <file number> / <dump-count>

CREATE can facilitate the creation of AND space for use in connection with the monitor Logical File Table. The parameter to CREATE is either:

<integer> CARDS, in which case a directory entry of <integer> cards will be created for the currently selected (USER, FILE) pair.

or

<integer> BLOCKS, in which case a directory entry of <integer> blocks will be CREATEd, and the number of card images will be set to the maximum number for <integer> number of blocks.

This instruction will not affect the scratch pointer, but a semi-DUMP will occur, that is, everything that DUMP does will occur except the actual scratch to record dump. Only the 320-word block containing the AND END-OF-FILE MARK, i.e., the last block, will be written out. Another parameter may follow the required <integer> CARDS or <integer> BLOCKS parameter.

EXAMPLE:

To create a directory entry of 400 cards, the following instruction should be executed:

CREATE 400 CARDS;

To create a directory entry of 9 blocks, the following instruction should be executed:

CREATE 9 BLOCKS;

F. Miscellaneous Instructions.

(22) ... AVAILABLE (no parameters)

The instruction AVAILABLE points out the total number of available blocks and the greatest number of consecutive available blocks, i.e., the greatest glob of available space. Thus, an execution of the instruction AVAILABLE will produce output similar to the following:

1436 AVAILABLE BLOCKS
GREATEST GLOB IS 140

The instruction DIRECTORY will automatically print the total number of available blocks and the greatest glob. This allows the user to make sure that a sufficiently large set of consecutive AND blocks is available when he wishes to DUMP a large set of images. Each AND block holds 15.24 (or $\frac{320}{21}$) images.

(23) ...BINARY (no parameters)

Sets a switch which inhibits attaching new serial numbers to the images entered into the Scratch Area. BINARY has two purposes:

(a) AND can be used to store files of binary information created as output in the Scratch Area by some other system. Subsequently, an AND program can fetch the binary information back into the Scratch Area in the normal manner, in groups of 21 G-20 32-bit logical words. A BINARY instruction should be given at the start of the AND program which fetches the information back into SCRATCH so that every 21st word will not be clobbered by a new serial number σ.

(b) If a BINARY instruction is given before AND instructions which edit normal alphabetic text, the original serial numbers will be preserved in Scratch. In particular, a card image fetched into the Scratch

Area from an AND record will have its original record serial number attached; an image read by TEXT will have a blank serial number. It is recommended that you do not do a DUMP of this information, since the DUMPed record will not have correct consecutive serial numbers.

Every FILE instruction turns the BINARY switch off.

(24) ... COMPARE (no parameter)

COMPARE will perform an image-by-image comparison of the images currently in the AND Scratch Area with the images in the currently selected AND record, checking for 1-to-1 identity of the images exclusive of their serial numbers.

(a) The COMPARE operation always begins by comparing the first image in scratch with the image at which δ is pointing.

(b) Each pair of images which are not identical will be printed along with their respective serial numbers.

(c) The comparison will cease when any of the following conditions is met:

(1) The last image currently in the Scratch Area has been compared;

(2) The End-of-File ("$$") image in the AND record is encountered; or

(3) More than ten pairs of images have been found to be non-identical and have been printed.

In any case, COMPARE will restore σ to its previous value.

The images which are not identical will be typed at a remote station if the program originated there even if no TTYPE instruction has been given, and will be printed on-line regardless of the print option.

Lack of identity is not considered an AND error and will not terminate the current run. Therefore,

the instruction CHECK has been provided to allow the user to make subsequent AND operations contingent upon the identity of two records.

(25) ... CHECK (no parameters)

If any COMPARE instruction has discovered a lack of identity, then CHECK will act like a DONE instruction. If no COMPARE has been given, or if all COMPARE's have found identity, then CHECK has no effect.

(26) ... DIRECTORY (no parameters)
(may have optional parameters) See Section G.

Lists the AND Directory entries for all records filed under the usage number of the most recent USER instruction. The following information is listed on one line for each record:

1. program number and dump count
2. number of card images
3. first AND block and total number of AND blocks occupied by the record on disc or on tape.

The usage number, current date, and the amount of available AND record space are also printed by DIRECTORY.

If the program originated from a remote teletype, then the directory will be typed out on the remote regardless of what print option is in force, whether or not a TTYPE instruction has been given.

(27) ... DONE (no parameters)

Terminate AND run. Writes an AND End-of-File image after the last image in the Scratch Area (See discussion in DUMP).

(28) ... PURGE (no parameters)

PURGE (parameters) (see Section G)

Remove the AND record selected by the most recent (USER, FILE) pair from the AND Directory, if this AND record has a Usage Number which agrees with that on the Job Card. The space which this record occupied on the disc will be made available to other AND records. PURGE marks <file number> undefined, so that a new FILE instruction must be given to subsequently refer to any AND record.

(29) ... RESET [n]

Sets Scratch Area pointer $\sigma \leftarrow n$. This instruction is normally used with n = 1 to clear the Scratch Area to start a new collation.

(30) ... DOLDUMP (no parameters)

DOLDUMP (parameters)

The often used sequence of instructions:

```
GET TO $ ;    DUMP ;
```

can be replaced by the instruction DOLDUMP. This will be very useful to programmers in the habit of forgetting

```
to              GETTO$ ;
before they     DUMP ;
```

DOLDUMP may have an optional parameter similar to that for DUMP.

(31) ... LAF '<two characters>'

The two characters enclosed between the quotes will be made the terminating language field for TEXT instructions and editing instructions which call on TEXT. The normal language field is AN so that a LAF 'AN' is effectively executed before AND starts processing instruction cards.

(32) ... SECRET

(33) ... PUBLIC

Provisions exist for permitting only the Man Number under which a program is filed to access the file. The instruction SECRET will mark the currently selected (USER, FILE) pair so that it can only be accessed if the Man Number on the job card agrees with the Man Number in the (USER, FILE) pair which has been marked as SECRET. The instruction PUBLIC will mark the currently selected (USER, FILE) pair so that it can be accessed by any Man Number. A file when first created will be accessable to any Man Number, i.e., in the PUBLIC state. (The instructions PUBLIC and SECRET may have a parameter(s) as described in Section G).

G. Optional Parameters Description

The instructions PURGE, SECRET, and PUBLIC may have an option <PARAMETER> as defined by the following:

<DUMP-COUNT> ::= /<integer> | /PERIL | <empty>

<PARAMETER> ::= <File Number> <Dump Count>

If the <DUMP COUNT> is <empty>, /PERIL will be used for the <DUMP-COUNT>. The effect is to make the <PARAMETER> the currently selected file for the execution of the instruction, as if a FILE <File Number>/<integer> instruction were given just previous to a PURGE, SECRET, or PUBLIC instruction. Thus the instructions

EXAMPLE 1:

```
PURGE 1/3;  PURGE 2/PERIL;  PURGE 3;
```

will accomplish the same result as the instructions

EXAMPLE 2:

```
FILE 1/3;  PURGE;
FILE 2/PERIL;  PURGE;
FILE 3/PERIL;  PURGE;
```

with the following exception:

Using the <PARAMETER> capabilities of PURGE (or SECRET OR PUBLIC), as in Example 1, will not abort later DUMPs or CREATEs because of files accessed PERILously, while an explicit FILE <File Number>/PERIL instruction, as in Example 2, will abort later DUMPs and CREATEs.

The <PARAMETER> to PURGE, SECRET, and PUBLIC may have a <PARAMETER LIST> as defined by

<PARAMETER LIST> ::= <PARAMETER> |
<PARAMETER LIST>, <PARAMETER>

thus the instructions of Example 1 may be written in one instruction, as in the following:

EXAMPLE 3:

PURGE 1/3, 1/PERIL, 3;

The <PARAMETER LIST> will be executed in left to right order, and an error on one item of the <PARAMETER LIST> will cause all items to the right of it to be ignored.

The instructions DUMP and CREATE may have an optional <PARAMETER> which is defined by the following:

<DUMP-COUNT> ::= / <integer>

<PARAMETER> ::= <FILE NUMBER> <DUMP-COUNT>

The effect is to mark the <PARAMETER> as the currently selected program, as if a

FILE <FILE NUMBER> <DUMP-COUNT>

were given just previous to the DUMP or CREATE instruction. Thus, the instruction

DUMP 7/0

is exactly equivalent to the instructions

FILE 7/0; DUMP;

The <PARAMETER> to CREATE is used in the following manner:

CREATE 71 CARDS FOR 9/0; CREATE 5 BLOCKS FOR 100/0;

The preceding instructions are exactly equivalent to the instructions

FILE 9/0; CREATE 71 CARDS;
FILE 10/0; CREATE 5 BLOCKS;
USER S228ZZ01
PEEK 100, ALL;

are executed, the first 100 cards (or if the record contains less than 100 cards, then the number of cards it contains) will be listed for all (USER, FILE) pairs containing the Man Number ZZ01.

Summary of AND Instructions

	Note: σ = Scratch Pointer	δ = Record Pointer
1.	USER <Usage Number>	: U ←<usage number> δ ←P←<undefined>
2.	FILE <File Number>/<dump count>	: P ←<program number> Select record (U , P); $\delta \leftarrow 1$; ¬ BINARY
3.	DUMP	: (Selected record) ←(Scratch Area)
	DUMP <File Number> /<dump count>	Select new record δ ←1; σ unchanged
4.	NOSAVE	
5.	RUN, <System Name> (or (<Logical File Type>))	
	RUN, <System Name> (or (<Logical File Type>)) , TAPE	
	RUN, <System Name> (or (<Logical File Type>)) , CARD	
	RUN, <System Name> (or (<Logical File Type>)) , (<Logical File Type>)	
6.	SET [p]	: $\delta \leftarrow p$
7.	GET [n]	: Fetch n images (δ), (δ + 1) . . . , (δ + n -1) into Scratch Area; $\delta \leftarrow \delta + n$; $\sigma \leftarrow \sigma + n$,
8.	GET TO [p]	: Fetch (p-δ +1) images (δ), (δ+1), .., (p) into Scratch Area; $\delta \leftarrow p + 1$; $\sigma \leftarrow \sigma + p - \delta + 1$
9.	TEXT	: Stop on 'AN' Read new

In any of the following, $ may be used for [q] or [r].

10.	LOAD [p] to [q]	≡	SET [p]; GET TO [q]
	LOAD [p]	≡	LOAD [p] TO [p]
11.	INSERT AFTER [q]	≡	GET TO [q] ; TEXT

12.	DELETE [p] TO [q]	≡	GET TO [p-1]; SET [q+1]
	DELETE [q]	≡	DELETE [q] TO [q]
13.	ALTER [p] TO [q]	≡	DELETE [p] TO [q] ; TEXT
	ALTER [q]	≡	DELETE [q]; TEXT
14.	PUT [p] TO [q] AFTER [r]	≡	GET TO [r]; LOAD [p] TO [q]; SET [r+1]
	PUT [q] AFTER [r]	≡	GET TO [r]; LOAD [q]; SET [r+1]

		AND Instructions?	new text?	from record?
15.	PRINT TEXT :	yes	yes	no
	PRINT ALL :	yes	yes	yes
	PRINT :	yes	no	no
	PRINT NO :	no	no	no
	PRINT SCRATCH	List entire present Scratch Area contents		
16.	PAGE	New Page		
17.	TTYPE [k] :	Give teletype output, k characters per lir		
	TTYPE ≡	TTYPE 66		
18.	NO TTYPE :	No teletype output		
19.	PEEK [n] :	Print n images from each record of currently selected user.		
20.	FILE <file number>/<dump count> (<integer>) :	Set FILE to be Logical File Type <integer		
21.	CREATE <integer> BLOCKS :	Reserve <integer> BLOCKS or CARDS of available space for currently selected (USER, FILE) pair		
	CREATE <integer> CARDS			
22.	AVAILABLE :	Print information about available space		
23.	BINARY :	No new serial numbers in Scratch Area		
24.	COMPARE :	Compare scratch to selected record		
25.	CHECK :	Terminate run if any errors have been found by COMPARE.		

26.	DIRECTORY	:	Print information about current users records.
27.	DONE	:	Terminate AND program
28.	PURGE	:	Delete currently selected file from directory
29.	RESET [n]	:	σ ←n
30.	DOLDUMP	≡	GET TO $; DUMP
31.	LAF '<two characters>'	:	Change text terminating language field to the <two characters> specified
32.	SECRET	:	Mark currently selected file accessable only to creator.
33.	PUBLIC	:	Mark currently selected file accessable to any user.

V. AND Error Messages

AND performs many checks on the validity of the instructions it executes. If an error is detected, AND will print:

(1) The current instruction card: (Note: except in print mode "NO", this will be the second time this card has been printed) ;

(2) A pointer of the form . . above the last column scanned on this instruction card, and

(3) An appropriate error message;

The following specific changes have been made:

(A) All superfluous <TAB> characters in AND instructions typed at a teletype are ignored in scanning AND instructions. The <TAB> character will still appear as '=' in the printed image of the instruction card.

(B) In the delimiter TO in parameters to AND editing instructions, the O ("OH") can be typed (by mistake) as a zero, without causing an AND error.

The error message can be classified as follows:

(1) Error of syntax on AND instruction card:

'IMPROPER AND INSTRUCTION'

'MISSING NUMERIC INSTRUCTION'

'IMPROPER CHARACTER ON INSTRUCTION CARD'

If columns 1 and 2 of an instruction card do not contain either 'AN' or blanks, AND will print the message 'IMPROPER LANGUAGE FIELD' and then attempt to scan columns 1 through 80 of the card instead of 3 through 30. If this modified scan yields well-formed AND instructions, they will be executed.

(2) Improper parameters to collation instruction.

The four messages which may occur and their meanings are summarized below; where there is a particular parameter in error, that parameter is denoted by "[e]". All of these four errors are recoverable.

(3) Improper parameters to other instructions:

'IMPROPER PROGRAM NUMBER' (P> 127)

'INVALID PRINT PARAMÉTER'

'IMPROPER PARAMETER TO RUN' (first parameter is not valid system name, or second parameter is not <empty>, 'CARD', 'TAPE', '<Logical File Type>')

'DUMP COÚNT INCORRECT'

'DUMP COUNT MISSING OR INCORRECT IN FORM'

(4) Miscellaneous

'USER OR PROGRAM MISSING'

'YOU ARE NOT PERMITTED TO DUMP, CREATE, OR PURGE THIS PROGRAM' (i.e., USER does not agree with job card)

'THAT AND RECORD HAS 0 CARDS' (a "RUN, <system>, TAPE" instruction has tried to set the |15 switch to take input from an AND record, or else a SET instruction has been given; the record contains no card images, possibly because an earlier AND run was terminated by an error before anything was dumped into the record).

'RUN TERMINATED BY CHECK'

'AND PROGRAM TERMINATED BY END-OF-FILE'

AND will generally recover from syntax errors on AND instruction cards. There are three cases:

(1) If an instruction has been scanned and a legal set of parameters follows, then the next item scanned must be a delimiter ... a semicolon, a bar, or the end of the card.

If not, AND will check that the character is a letter; if it is a letter, it will be assumed that there was a semi-colon omitted, the message:

MISSING DELIMITER

will be printed, and AND will start scanning the letter looking for a correct instruction. Any other case will fall under item (2).

(2) If a syntax error occurs in the parameter of an instruction, AND will skip to the next delimiter and print the message:

IMPROPER PARAMETER

(3) If an alphabetic instruction name is encountered that is not a legal AND instruction, the message:

IMPROPER 'AND' INSTRUCTION

will be printed, and AND will skip ahead to the next delimiter to start scanning again.

More than 20 AND errors will cause AND to terminate with the message:

TOO MANY ERRORS, I QUIT.

(4) AND System Errors

If any error message containing the phrase:

"(PANIC****)"

occurs, it should be brought to the immediate attention of Computer Center staff, since it indicates a potentially serious AND system error.

AND REFERENCE

Users are referred to User Consultant and Dennis Moyles (staff member), Director of Special Projects.