DEBUG AND TRACE - ALAN HEWER, DOUG JEUNG

42

DEBUG

- DEBUG IS A CALLABLE INTRINSIC WHICH ALLOWS NON-PRIV USERS TO DYNAMICALLY CHECKOUT A PROGRAM WHEN RUNNING IN A SESSION.
- IT IS ORIENTED TOWARD USERS WHO ARE KNOWLEDGEABLE ABOUT THE CODE AND DATA STRUCTURES WHICH CONSTITUTE THE EXECUTING PROGRAM.
- SIMPLE INTERACTIVE COMMANDS ALLOW USERS TO
    - DISPLAY / MODIFY REGISTER CONTENTS
    - DISPLAY / MODIFY DATA IN THE STACK
    - SET / RESET CODE BREAKPOINTS
- PROTECTION IS GUARANTEED.

<u>INVOCATION</u>

DEBUG IS ENTERED BY THE FOLLOWING MEANS

- DIRECT EXTERNAL CALL TO THE INTRINSIC DEBUG
  WITH EXTERNAL DECLARATION :

  PROCEDURE <u>DEBUG</u>; OPTION EXTERNAL;
- VIA AN EXISTING CODE BREAKPOINT.

NOTE THAT IT IS NECESSARY TO COMPILE ONLY ONE DIRECT CALL TO DEBUG WITHIN THE PROGRAM. CONTROL MAY BE TRANSFERRED TO DEBUG AT ALL OTHER LOCATIONS WITHIN THE PROGRAM UTILISING THE BREAKPOINT FACILITY.

ENTRY

A WELCOME MESSAGE IS OUTPUT OF THE FORM:

$$\left\{\begin{matrix}\text{DEBUG}\\ \text{BREAK}\end{matrix}\right\} \ \text{<segment>.<offset>}$$

DEBUG -	INDICATES A DIRECT CALL TO DEBUG
BREAK -	INDICATES A BREAKPOINT ENTRY
<segment> -	LOGICAL PROGRAM SEGMENT
<offset> -	LOCATION OF THE INSTRUCTION, IN ABOVE SEGMENT, TO BE EXECUTED UPON RETURN. IF 'BREAK', THE INSTRUCTION BREAKPOINTED IS NOT EXECUTED PRIOR TO ENTRY TO DEBUG.

<u>PROMPT</u> CHARACTER "?"

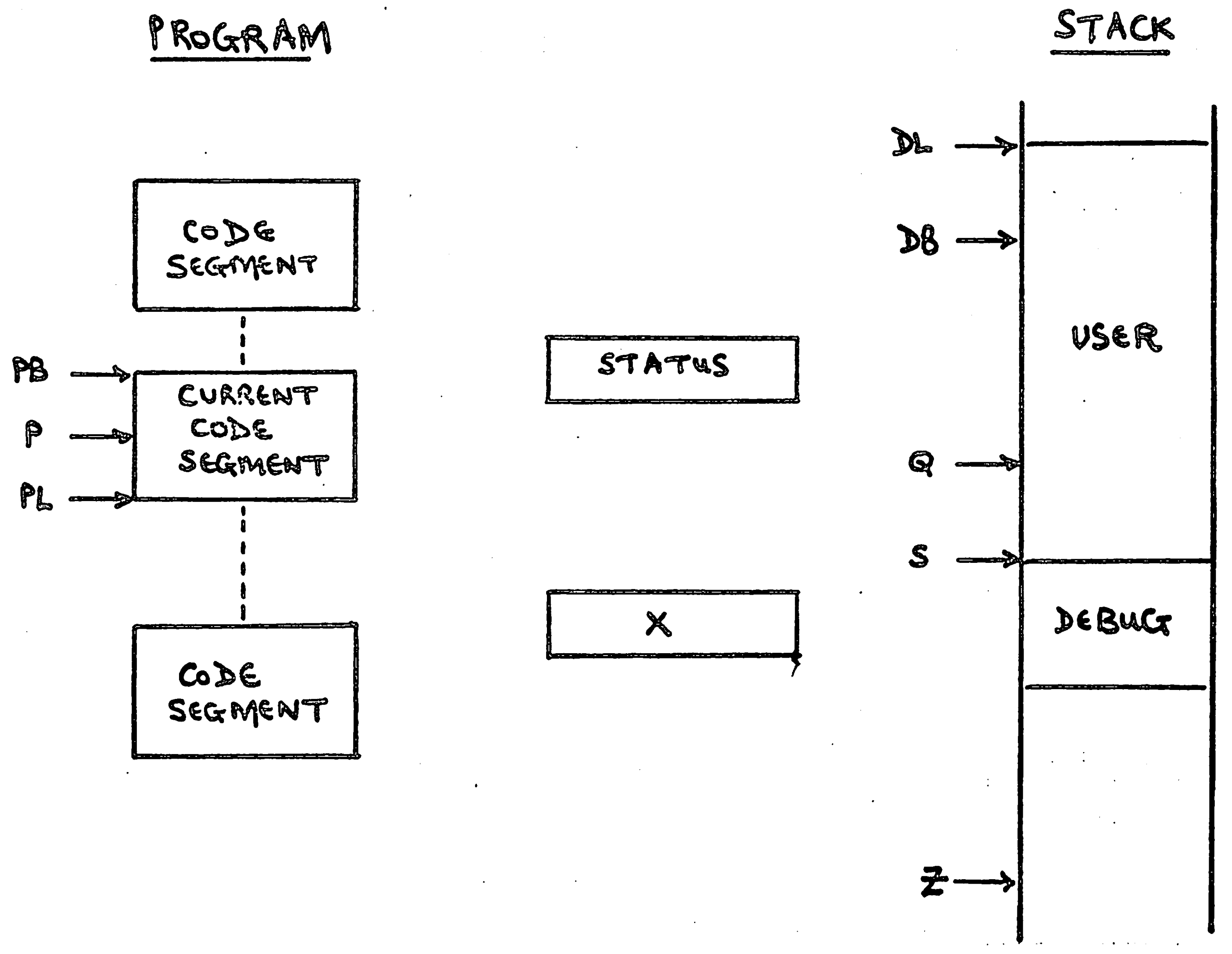
<u>COMMAND</u> - SINGLE LETTER COMMAND MNEMONIC
- PARAMETERS
- BLANKS IGNORED
- TERMINATED WITH CARRIAGE RETURN

<u>ERROR</u> MESSAGES

"NONO n"	- ILLEGAL SYNTAX OR INALID INFORMATION
"XTRA n"	- TOO MUCH INPUT ON A MODIFY MEMORY
"FULL n"	- BREAKPOINT TABLE FULL

n = CHAR POSITION AT WHICH ERROR DETECTED

4A


PROGRAM
CODE SEGMENT
PB
P
PL
CURRENT CODE SEGMENT
CODE SEGMENT
STATUS
X
STACK
DL
DB
Q
S
Z
USER
DEBUG

COMMAND SYNTAX

```
number     ::=  octaldigit .......                            {OCTAL ONLY
expr       ::=  [+|-] number [{+|-} number] .....
offset     ::=  expr                                          {(P-PB) ADDRESS
segment    ::=  expr
location   ::=  [segment .] offset                            {DEFAULT SEGMENT IS
                                                                      CURRENT
reg        ::=  Q | S | X | ST | P | Z | DL
count      ::=  expr
stackadr   ::=  [Q|S|DL] expr [I [expr]] ....                 { I = INDIRECTION
                                                              { DEFAULT BASE = DB
datalist   ::=  [expr] [, [expr]] ..... [.]                   {.  = LIST TERMINATOR
filename   ::=  anydigit .......
```

COMMANDS

D	[(filename)] [reg]	DISPLAY REGISTERS
M	reg = expr [, reg = expr]	MODIFY REGISTERS
D	[(filename)] stackadr [, count]	DISPLAY MEMORY (STACK)
M	stackadr [, count] [= datalist]	MODIFY MEMORY (")
T		TRACE STACK MARKERS
B	location [, location]	SET BREAKPOINTS
C	[location [, location] . . .]	CLEAR BREAKPOINTS
R	[location]	RESUME

DISPLAY REGISTERS D [(filename)] [reg].....

D	DISPLAY ALL REGISTERS
D S X Z	DISPLAY S, X, Z REGISTERS
D (L) ST	DISPLAY STATUS REG ON FILE L

IF A FILE IS SPECIFIED, DEV = LP IS TAKEN. THIS CAN BE OVERRIDDEN BY A :FILE COMMAND

MODIFY REGISTERS M reg = expr [, reg = expr]......

M X = 4	SET X TO 4
M Q = 50 , S = 160	SET Q TO 50 , S TO 160
M P = 3.66	(SPECIAL CASE) SET P TO LOCATION 66 OF LOGICAL SEGMENT 3.

NOTE THAT DL <= Ø <= Q <= S <= Z.

ONLY BITS 2 THRU 7 OF STATUS MAY BE CHANGED.

DISPLAY MEMORY   D [(filename)] stackadr [, count]

D 25	DISPLAY LOCATION DB+25
D Q+6I, 5	DISPLAY 5 WORDS POINTED TO BY LOCATION Q+6
D (LP) Q+6I, 5	SAME AS ABOVE, BUT ON FILE LP

MODIFY MEMORY   M stackadr [, count] [= datalist]

M S-2 = 555	CONTENTS OF LOCATION S-2 SET TO 555
M DL+5I, 20	MODIFY 20 WORDS OF LOCATIONS POINTED TO BY DL+5. ASSUME THIS ADDRESS IS DB+24 THEN: DEBUG OUTPUTS THE ADDRESS PLUS CONTENTS, THE USER INSERTS CONSECUTIVE NEW CONTENTS A BLANK LEAVES UNCHANGED

```
+000024 = 4, 1,2,,,5
+000031 = ɸ, ,,10.
```

AS IN LOCATIONS 26,27,31,32 ABOVE. UP TO 20 WORDS WERE SPECIFIED, BUT A PERIOD "." ACTS AS A TERMINATOR

TRACE STACK   T

```
T          PRINTS A TRACE OF USER STACK MARKERS
            SHOWING  LOGICAL SEGMENT # AND  P LOCATION.
7.50       (CURRENT LOCATION)
4.65       (LOC FROM WHICH 7.50 WAS CALLED)
0.10       (LOC  "   "    4.65  "   "  )
0.24       (LOC  "   "    0.10  "   "  . THIS IS THE
             LAST STACK MARKER FOUND AND SHOULD BE IN
             OUTER BLOCK OF PROGRAM)
```

SET BREAKPOINT B location [, location] .....

B 24	SET BP AT LOC 24 OF CURRENT SEGMENT
B 3.10, 16	SET BP'S AT LOC 10 OF SEG 3, AND 16 IN CURRENT SEGMENT.

CLEAR BREAKPOINT C [location [, location] ....]

C 24, 3.10	CLEAR BREAKPOINTS ORIGINALLY SET IN LOC 24 OF CURRENT SEG, 10 IN SEG 3.
C	CLEAR ALL BREAKPOINTS

RESUME R [location]

R	RETURN TO USER PROGRAM AT EITHER INSTRUCTION FOLLOWING DIRECT CALL TO DEBUG OR INSTRUCTION WHICH GENERATED BREAK.
R 5.33	SET BREAKPOINT AT 5.33 AND THEN RETURN.

BREAKPOINT MECHANISM

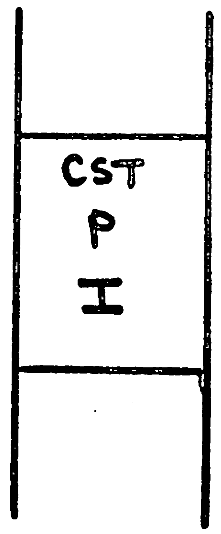
SET BP AT LOCATION LCST. P. ASSUME VALID.
CONVERTS LCST (logical) TO CST (physical).
ENTER CST AND P IN BREAKPOINT TABLE
IS SEGMENT PRESENT IN MEMORY ?

No — DO NOTHING

Yes — GET INSTRUCTION I
STORE IN BP TABLE
REPLACE WITH %30000

WHENEVER A SEGMENT IS BROUGHT INTO MEMORY
IT IS CHECKED FOR A POTENTIAL BREAKPOINT.
IF SO, THE INSTRUCTION I IS STORED IN BP TABLE
AND REPLACED WITH %30000.
THUS, WHENEVER A SEGMENT IS PRESENT IN MEMORY,
IT CONTAINS ALL BREAKPOINTS WHICH ARE OUTSTANDING.

BREAKPOINT TABLE



DURING EXECUTION OF THE SEGMENT. HITTING THE
%30000 PROVOKES AN ILLEGAL INSTRUCTION TRAP
(INTERNAL INTERRUPT 15).
CHECK BP TABLE FOR VALID BREAKPOINT.
IF NOT ... ABORT USER
IF YES ... CALL DEBUG.

```
                |
                WELCOME MESSAGE " BREAK LCST.P "
                |
                REPLACE INSTRUCTION I IN SEGMENT
                |
                CLEAR ENTRY IN BP TABLE
                |
                etc .......
```

TRACE

- SYMBOL ORIENTED DEBUGGING TOOL
  - AVAILABLE WITH FORTRAN AND SPL
  - REQUIRES COMPILE TIME COMMANDS TO SET UP TABLE OF IDENTIFIERS TO BE TRACED
- DOES NOT REQUIRE USER TO BE KNOWLEDGEABLE ABOUT CODE AND DATA STRUCTURES
- SIMPLE INTERACTIVE COMMANDS ALLOW USERS TO
  - DISPLAY / MODIFY DATA
  - LIST SELECTIVELY THE CONTENTS OF VARIABLES WHICH HAVE BEEN MODIFIED
  - RETURN CONTROL TO USER AT SELECTED POINTS

<u>USER ACTION AT COMPILE TIME</u>

- USER MUST PRECEDE SOURCE DATA TO COMPILER WITH

  $ TRACE <program unit>; <identifier list>

  <program unit> IS NAME OF MAIN PROGRAM OR SUBROUTINE (FORTRAN), PROCEDURE (SPL)

  <identifier list> NAMES OF

  VARIABLES
  ARRAYS
  LABELS
  ROUTINES

- ACTION BY COMPILER AS A RESULT OF ABOVE
  - BUILD (COMPILE TIME) A TABLE WHICH RESIDES BEHIND THE USER'S SECONDARY DB AT RUN TIME
  - GENERATE CALLS TO LIBRARY ROUTINE TRACE1'

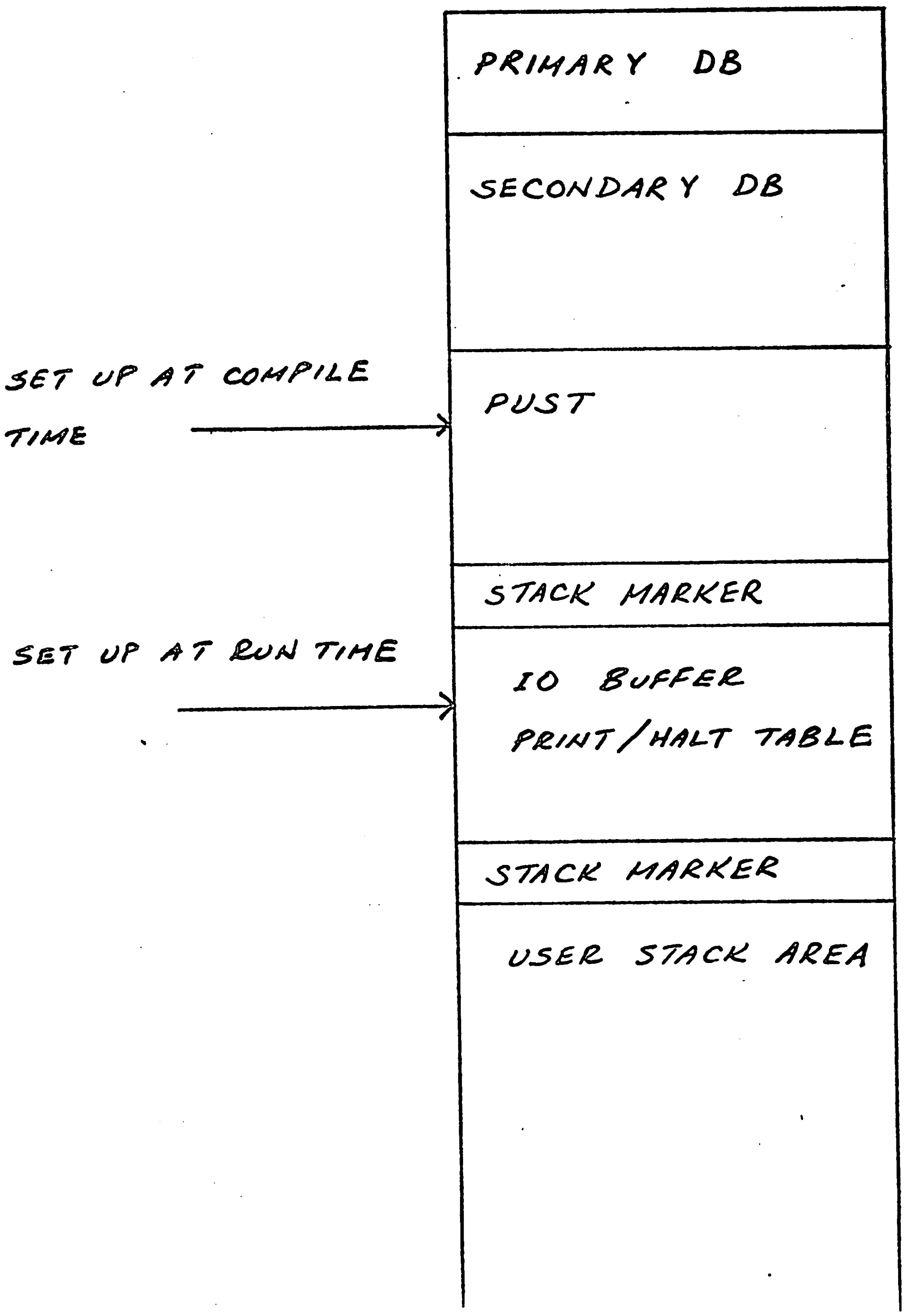
CALLING SEQUENCES

CONDITION CODE					
PUST OF WHICH THIS ENTRY IS A PART					
I′	X′	TS	M	S	LOCATION OF ENTRY IN PUST

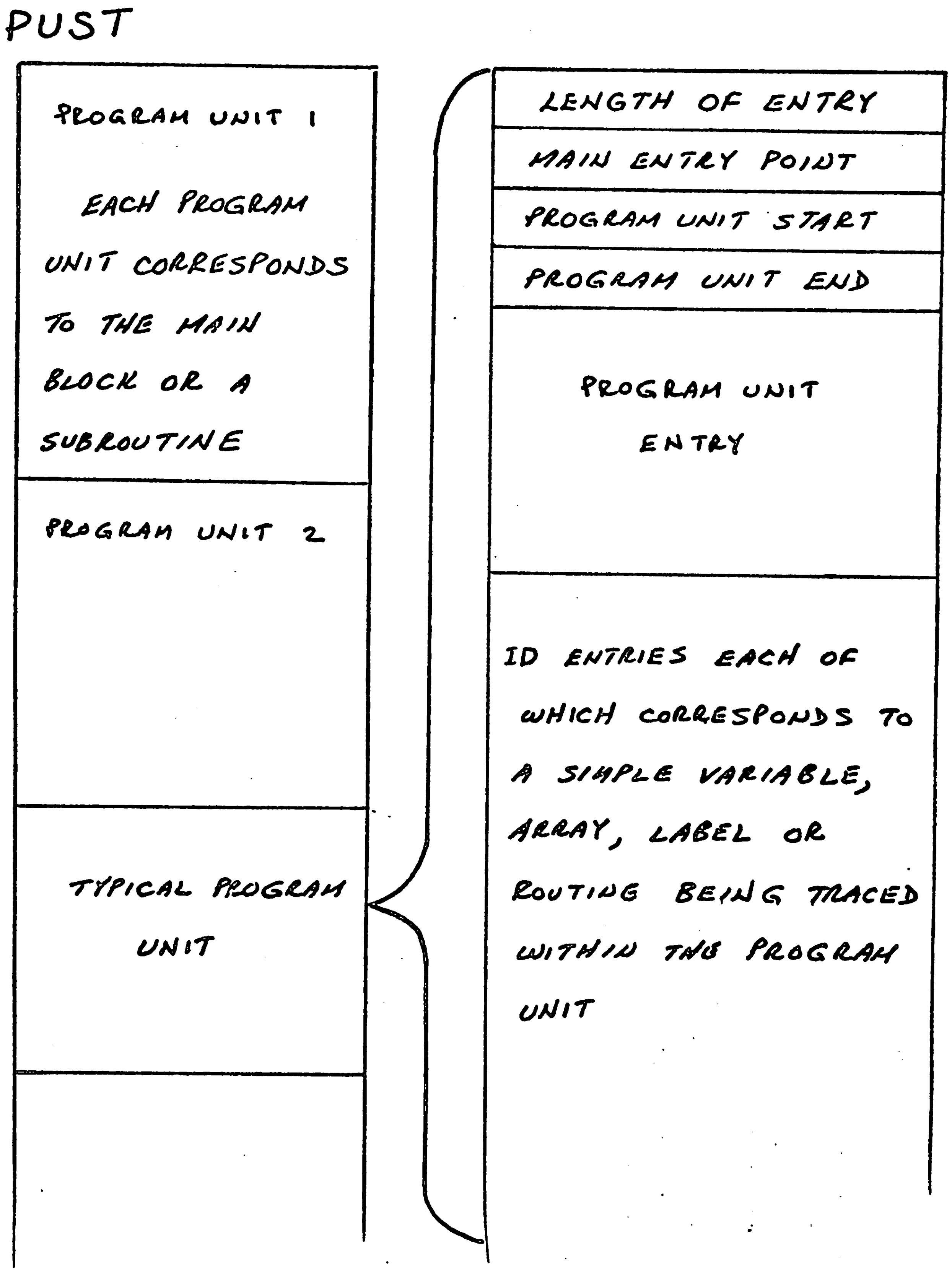
I′	= 1	DIRECT
X′	= 1	NOT INDEXED
TS	= 11	ENTERING A TRACED ROUTINE
	00	CALLING A TRACED ROUTINE
	01	RETURNING FROM A TRACED ROUTINE
	10	EXITING FROM A TRACED ROUTINE
M	= 1	USE TS (ROUTINE)
	0	VARIABLE, ARRAY OR LABEL
S	= 1	SPL
	0	FORTRAN

④

USER DATA AREA



PUST TABLE ENTRY


PUST
PROGRAM UNIT 1
EACH PROGRAM UNIT CORRESPONDS TO THE MAIN BLOCK OR A SUBROUTINE
PROGRAM UNIT 2
TYPICAL PROGRAM UNIT
LENGTH OF ENTRY
MAIN ENTRY POINT
PROGRAM UNIT START
PROGRAM UNIT END
PROGRAM UNIT ENTRY
ID ENTRIES EACH OF WHICH CORRESPONDS TO A SIMPLE VARIABLE, ARRAY, LABEL OR ROUTINE BEING TRACED WITHIN THE PROGRAM UNIT

FORMAT OF ID ENTRIES

CLASS		TYPE	ENTRY LENGTH
# OF CHARACTERS		CHAR 1	
CHAR 2		CHAR 3	
CHAR 4		CHAR 5	
CHAR 6		ETC	

CLASS	=	00001	SIMPLE VARIABLE
		00010	ARRAY
		10000	LABEL
		01000	SUBROUTINE
		01100	FUNCTION ROUTINE
TYPE	=	0000	LOGICAL
		0001	INTEGER
		0010	REAL
		0011	DOUBLE
		0100	STRING
		0101	LONG
		0110	COMPLEX

F

FORMAT OF ID ENTRIES (CONT.)

LABEL :

SIMPLE VAR :

ARRAY :

	MODE	I	X	DISPLACEMENT

MODE	=	000	DB +
		010	Q +
		011	Q −
		101	S −

ROUTINE :

etc.	P_3
P_2	P_1
#PARAMETERS	#WORDS IN STACK

P_i L M TYPE

L	=	00	USE M AND T
		01	SPL LABEL PASSED AS PARAMETER FORTRAN POINTER TO ARRAY

M	=	00	VALUE
		01	REFERENCE

USER ACTION AT RUN TIME

- COMMANDS WHICH MAY BE USED
  - PRINT
  - HALT
  - GO
  - SET
  - DROP

- IDENTIFIERS WHICH MAY BE TRACED
  - SIMPLE VARIABLE
  - ARRAY
  - LABEL
  - ROUTINE

GO COMMAND

PROMPT CHARACTER "*"

USER TYPES IN

GO	EXECUTION CONTINUES
GO <label name>	EXECUTION CONTINUES FROM LABEL LOCATION WITHIN PROGRAM UNIT

PRINT COMMAND (HALT)

USER TYPES IN

```
      PRINT <program unit>
      <identifier>  [<conditions>]
      <identifier>  [<conditions>]
      ---
      ----
      BLANK RECORD
```

PROMPT CHARACTER IS AGAIN EMITTED BY TRACE UNTIL A "GO" COMMAND IS ENTERED

<program unit>	NAME OF MAIN PROGRAM OR ROUTINE
<identifier>	VARIABLE, ARRAY, LABEL OR ROUTINE

PRINT	RESULTS IN IDENTIFIER AND VALUE IF ANY TO BE PRINTED
HALT	SAME AS PRINT AND IN ADDITION CONTROL RETURNED TO USER

- CONDITIONS C1, C2, C3 AND C4
    - NOT REQUIRED
    - IF PRESENT MUST BE IN ORDER
    - C1 SUBSCRIPT FOR ARRAYS

      ARR (* =)

      INTEGER CONSTANT OR INTEGER SIMPLE VARIABLE

    - C2 VALUE

      I >

      CONSTANT OR VARIABLE OF SAME TYPE AS "I".

      ARR (* : 24) <=

    - C3 BOUNDS

      I L1 - L2 MUST BE LABELS IN PROGRAM UNIT

    - C4 COUNT

      I @5 EVERY FIFTH OCCURRENCE

SET COMMAND

TRACE OUTPUT IS UNDERLINED IN FOLLOWING EXAMPLE

```
* SET PROGI cr                     cr = CARRIAGE RETURN
 BOB = 31 cr                      PRINTS VALUE
 ARR (42) = 22 /33 cr             PRINTS VALUE WHICH MAY BE
                                      CHANGED

 ARR (42), 3 = 33  14  5          BLOCK OUTPUT
 DB + 5 = 31                      REGISTER REFERENCE ALLOWED
```

DROP COMMAND

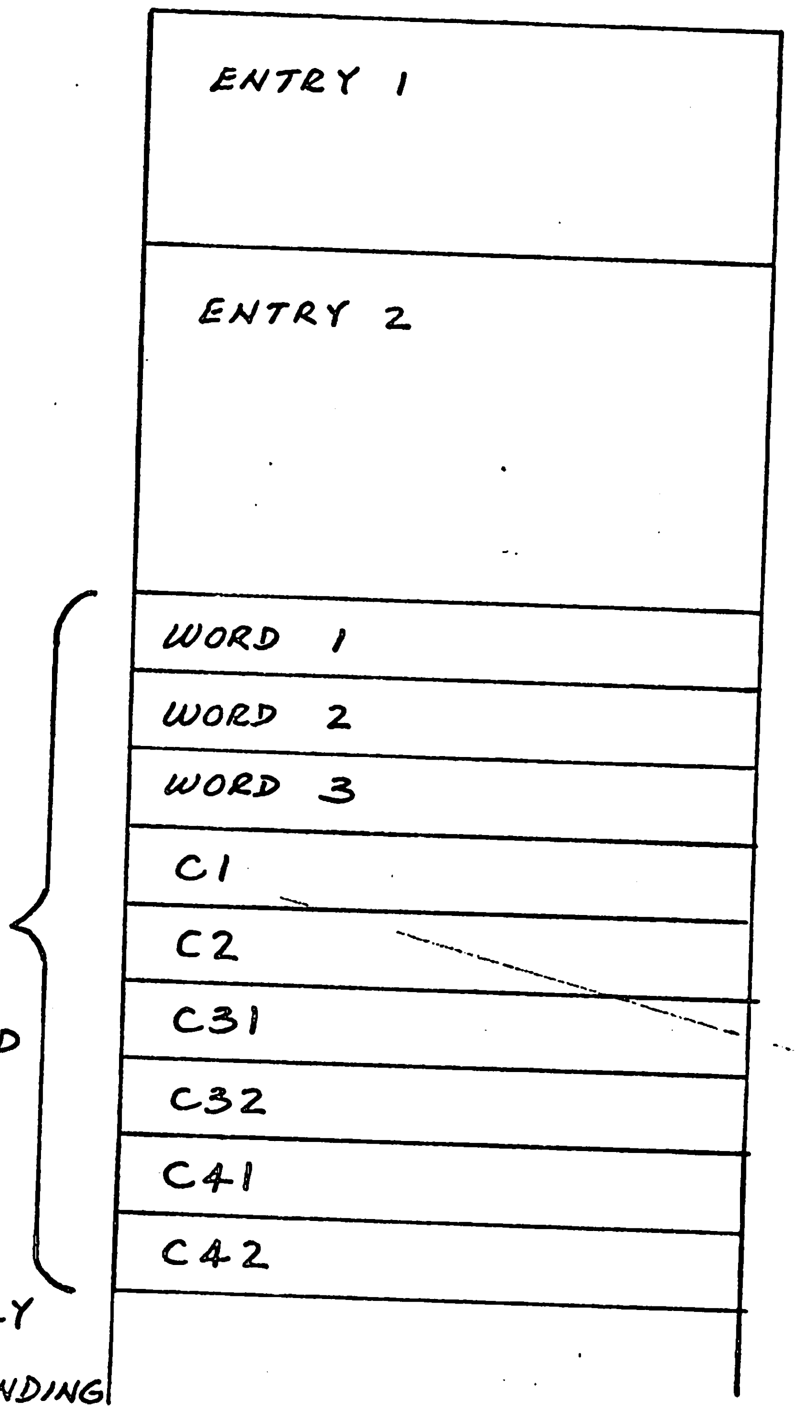
```
DROP <program unit>            SELECTIVELY REMOVES ENTRIES
<identifier>                   FROM PRINT/HALT TABLE
<identifier>
- - - -
- - - -
BLANK RECORD


DROP <program unit>            REMOVES ALL ENTRIES
$ ALL                          BELONGING TO PROGRAM UNIT

DROPALL                        REMOVES ALL ENTRIES
```

14

PRINT / HALT TABLE



TYPICAL ENTRY

WORDS 1-3 ALWAYS REQUIRED

C1 - C41 ARE OPTIONAL AND ARE REQUIRED ONLY IF THE CORRESPONDING CONDITION EXISTS

PRINT / HALT TABLE ENTRY

WORD 1	LENGTH OF ENTRY
WORD 2	POINTER TO BASE OF PUST
WORD 3	

0 2	3	4 6	7	8	9	10			13		15
R1	T1	R2	T2	E3	E4	T4			H		PA

CONDITION 1	R1	=	000	NO CONDITION 1
			001	<
			010	=
			011	<=
			100	>
			101	<>
			110	>=
	T1	=	1	C1 IS ADDRESS
			0	C1 IS A CONSTANT

CONDITION 2 R2, T2 CORRESPOND TO R1, T1

CONDITION 3	E3 = 1	CONDITION 3 EXISTS
	C31	LOWER BOUND
	C32	UPPER BOUND
CONDITION 4	E4 = 1	CONDITION 4 EXISTS
	T4 = 1	C42 IS ADDRESS
	0	C42 IS A CONSTANT
	C41	COUNTER

H = 1 HALT ENTRY OTHERWISE PRINT ENTRY

PA = 1 FOR PROCEDURES ONLY, PRINTS PARAMETERS

CONTROL Y

- RETURNS CONTROL TO USER EVEN IF NO "HALT" HAS BEEN ENTERED IN TABLE.
    - IF EXECUTING IN A TRACED PROGRAM UNIT THEN A MESSAGE INFORMING USER OF THE PROGRAM UNIT IS EMITTED, FOLLOWED BY THE PROMPT CHARACTER "*"
    - IF NOT IN A TRACED PROGRAM UNIT THEN CONTROL IS RETURNED TO USER AS SOON AS A TRACED PROGRAM UNIT IS ENTERED.