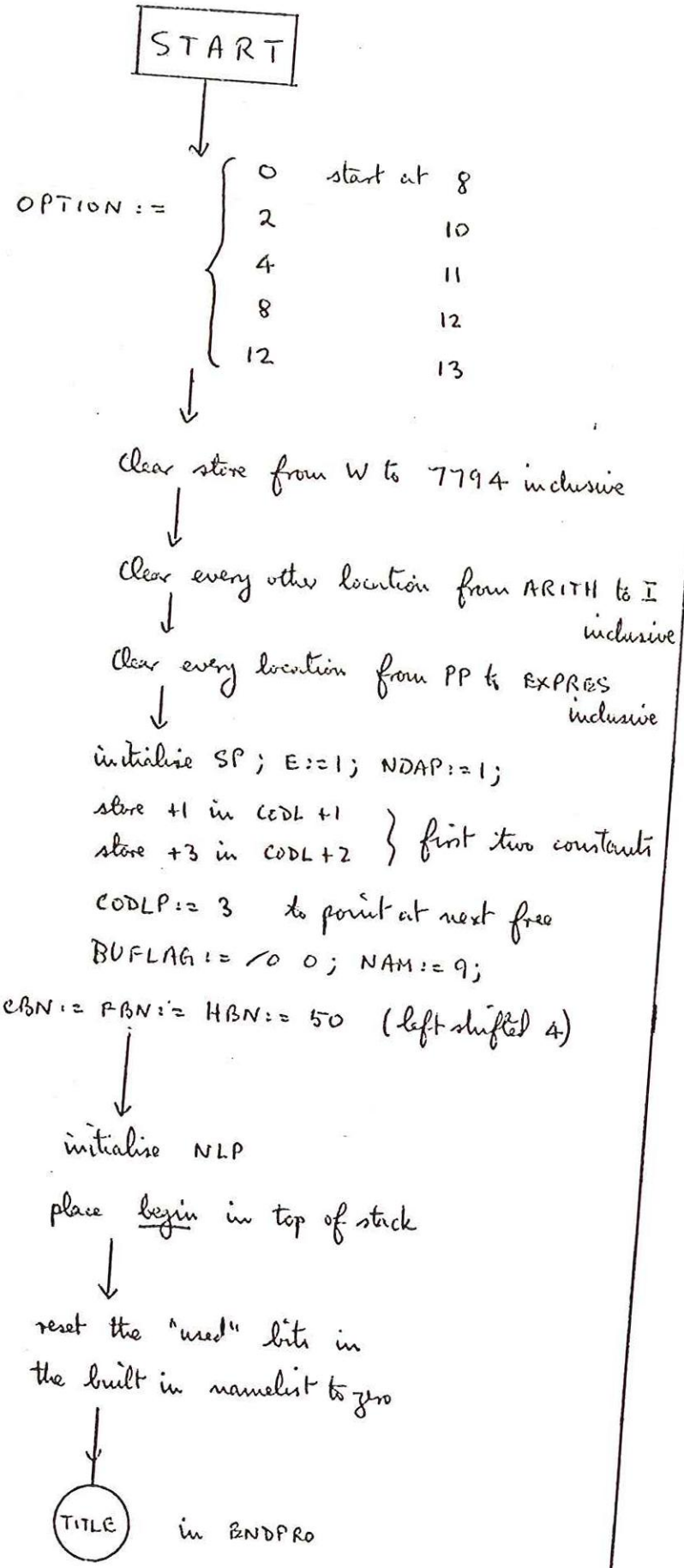
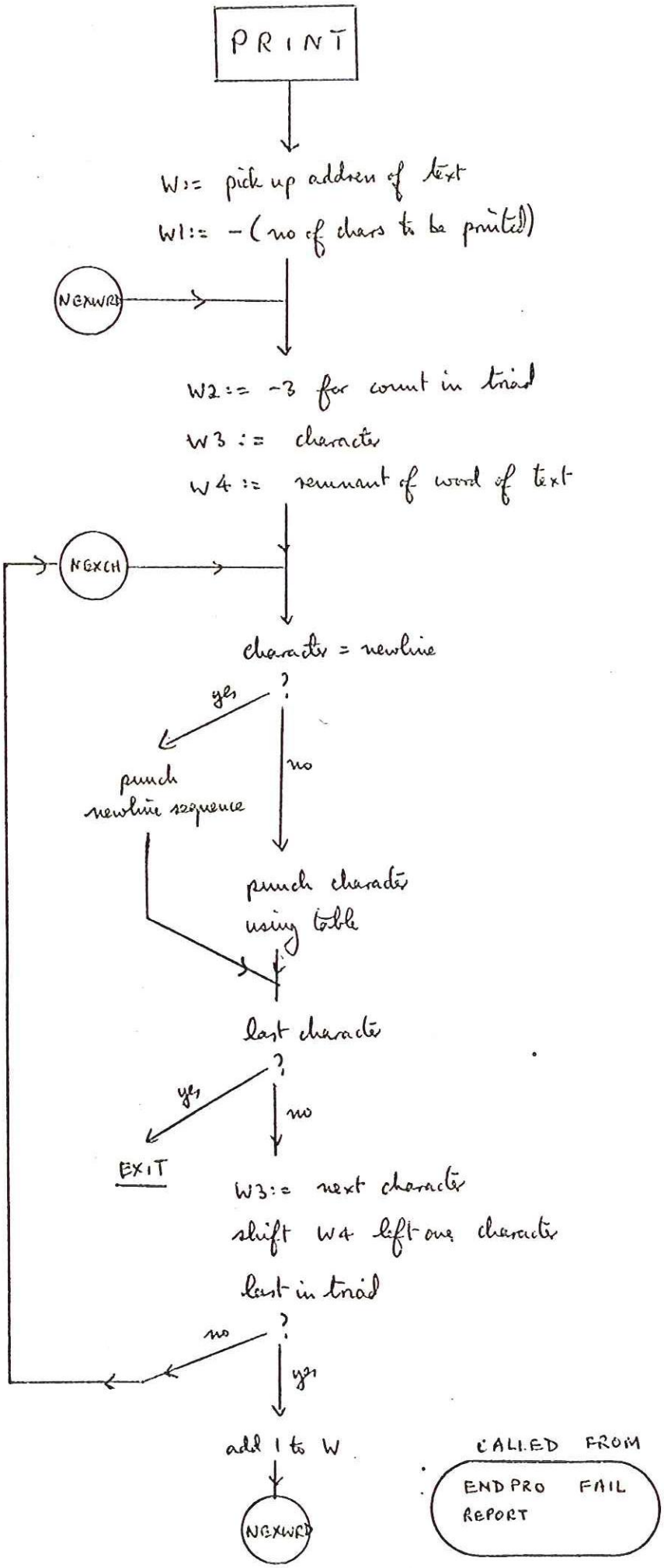
ELLIOTT 903 ALGOL

Index to flowcharts - December 1966

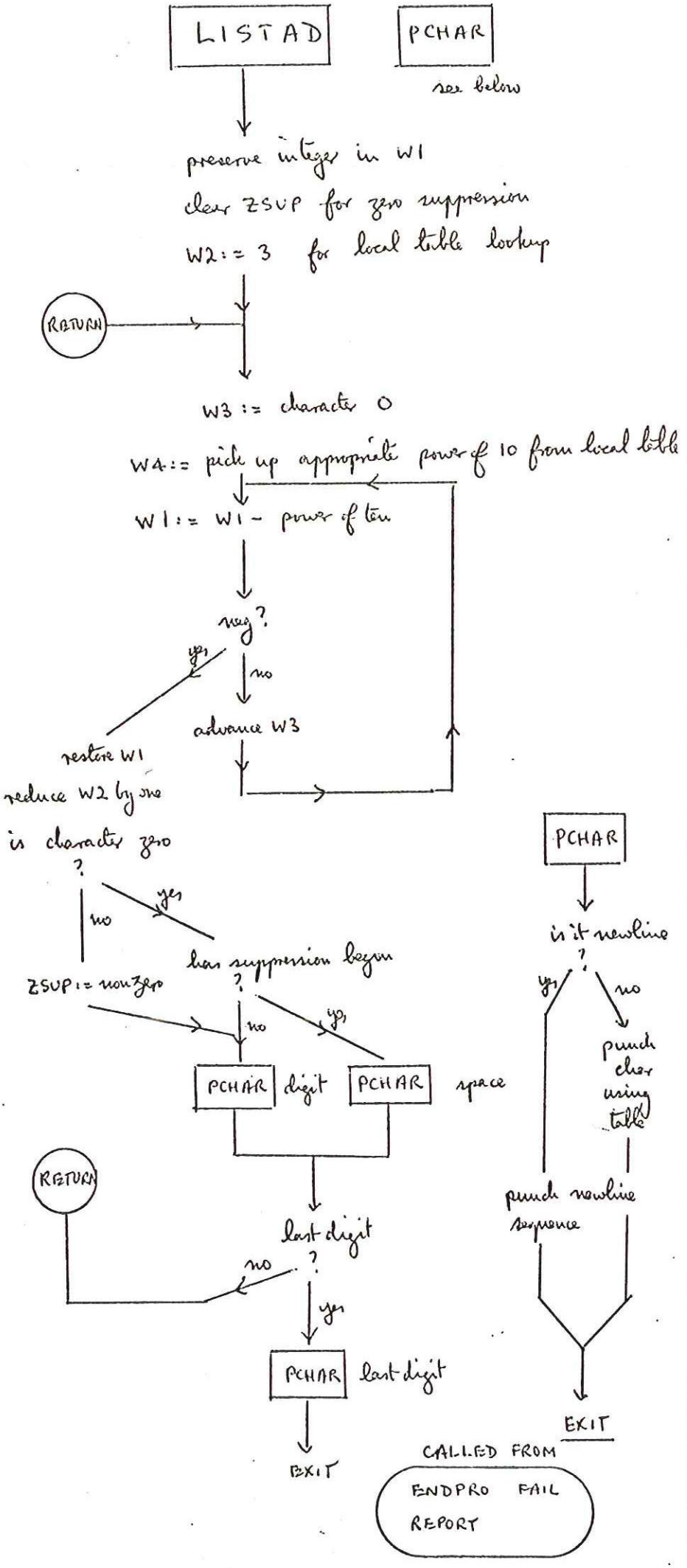
The flowcharts are given in the order in which they appear in the listing.

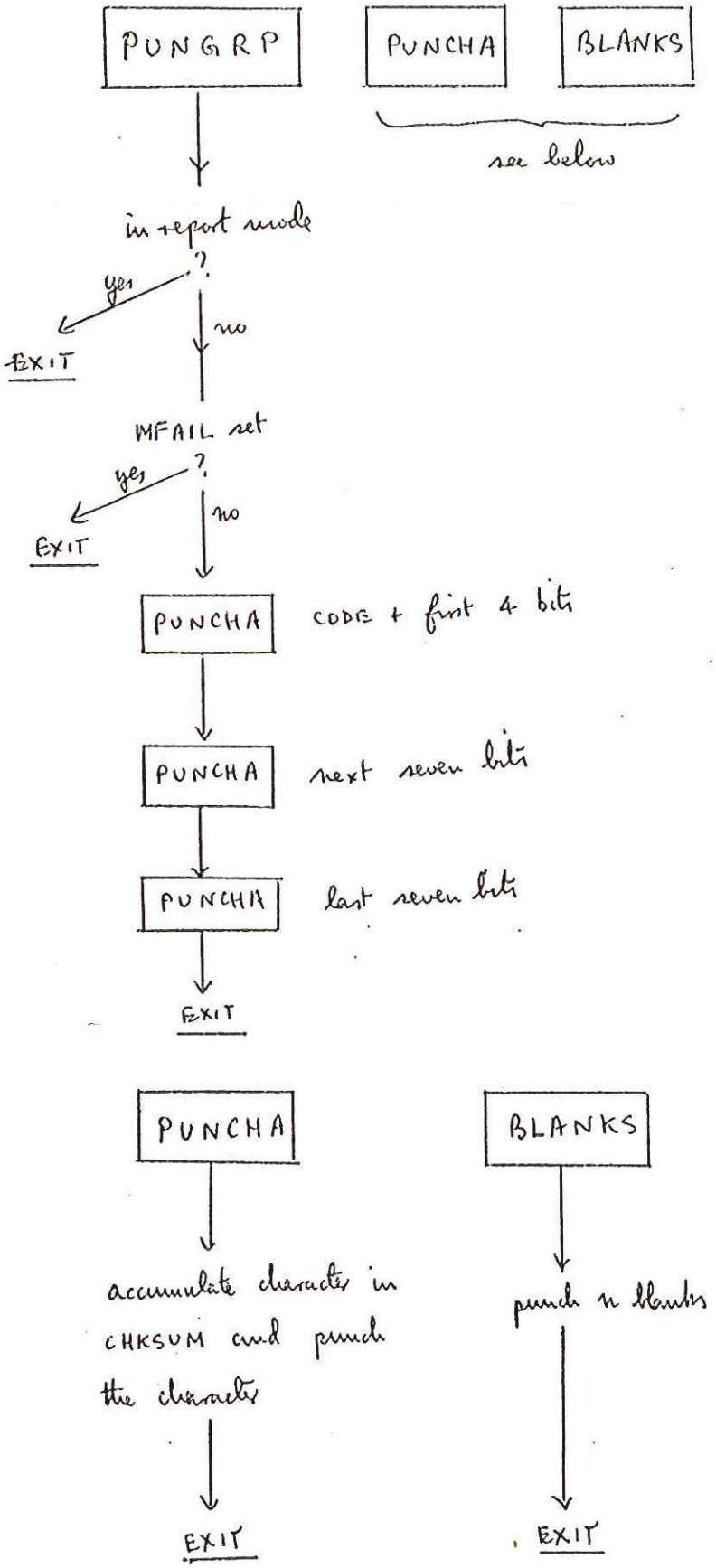

START
OPTION :=
0 start at 8
2 10
4 11
8 12
12 13
Clear store from W to 7794 inclusive
Clear every other location from ARITH to I inclusive
Clear every location from PP to EXPRES inclusive
initialise SP ; E:=1; NDAP:=1;
store +1 in CODL +1
store +3 in CODL +2
first two constants
CODLP:= 3 to point at next free
BUFLAG:= 0 0 ; NAM:= 9;
CBN:= FBN:= HBN:= 50 (left shifted 4)
initialise NLP
place begin in top of stack
reset the "used" bits in the built in namelist to zero
TITLE
in ENDPRO

print text


PRINT
W:= pick up address of text
W1:= -(no of chars to be printed)
NEXWRD
W2:= -3 for count in triad
W3 := character
W4 := remnant of word of text
NEXCH
character = newline
?
yes
no
punch newline sequence
punch character using table
last character
?
yes
no
EXIT
W3:= next character
shift W4 left one character
last in triad
?
no
yes
add 1 to W
NEXWRD
CALLED FROM
END PRO FAIL
REPORT

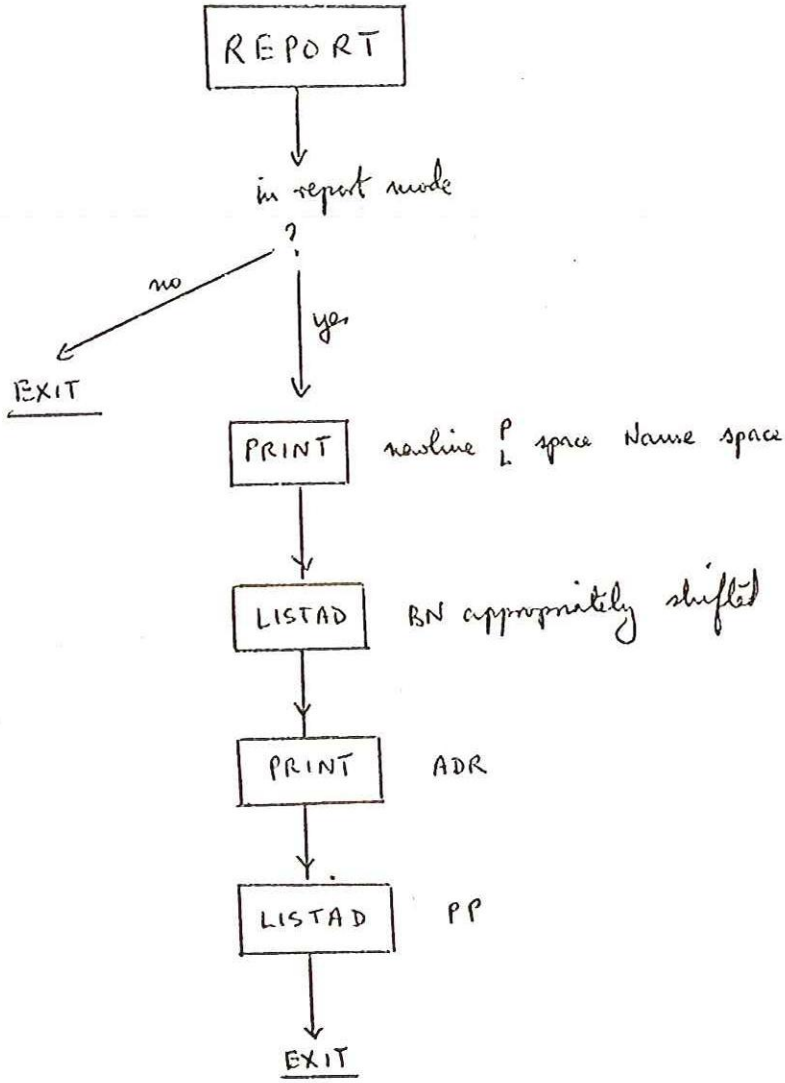
Convert binary integer and print it





punch a rel-bin output word

CALLED FROM
RRBRAK COMPIL
FOMPIL UPDATE
ENDPRO

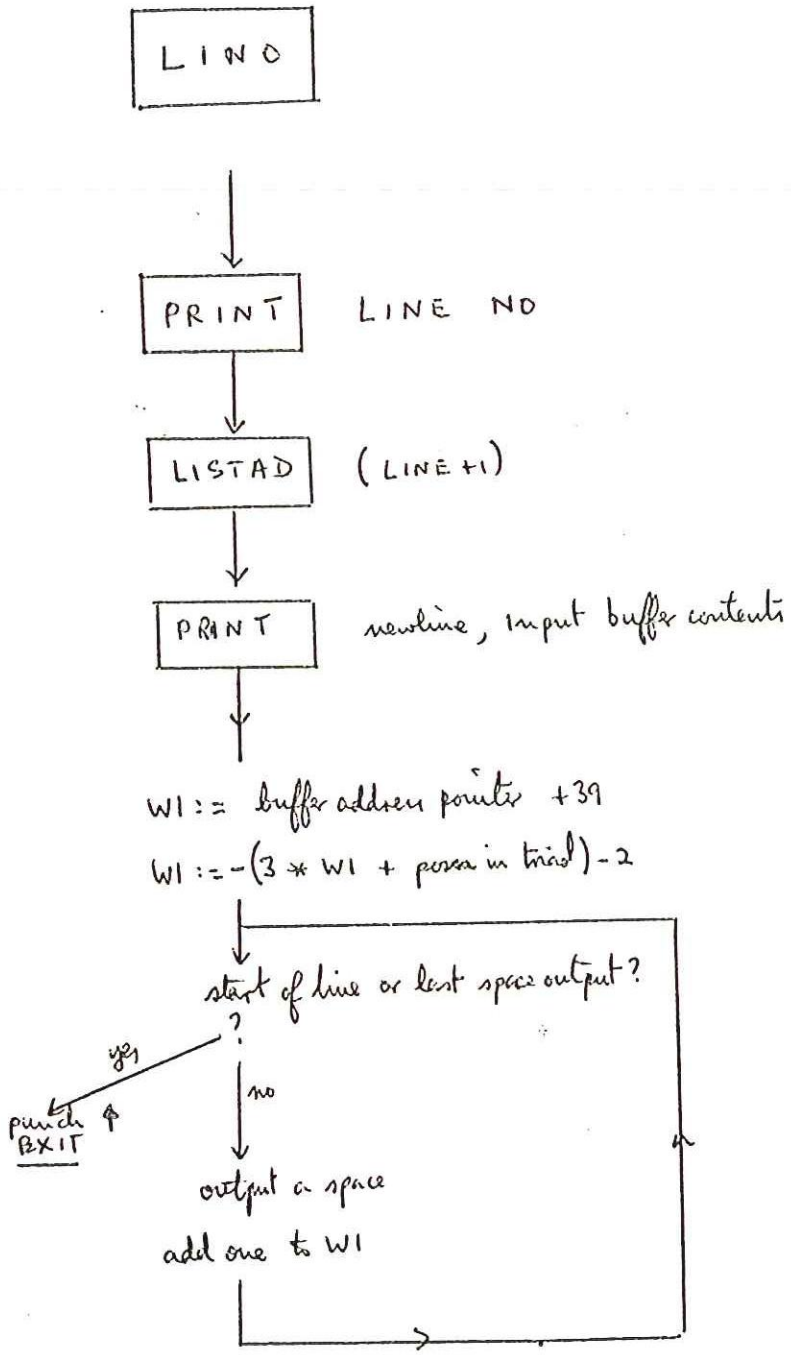


Reports occurrence of labels and procedures.

CALLED FROM

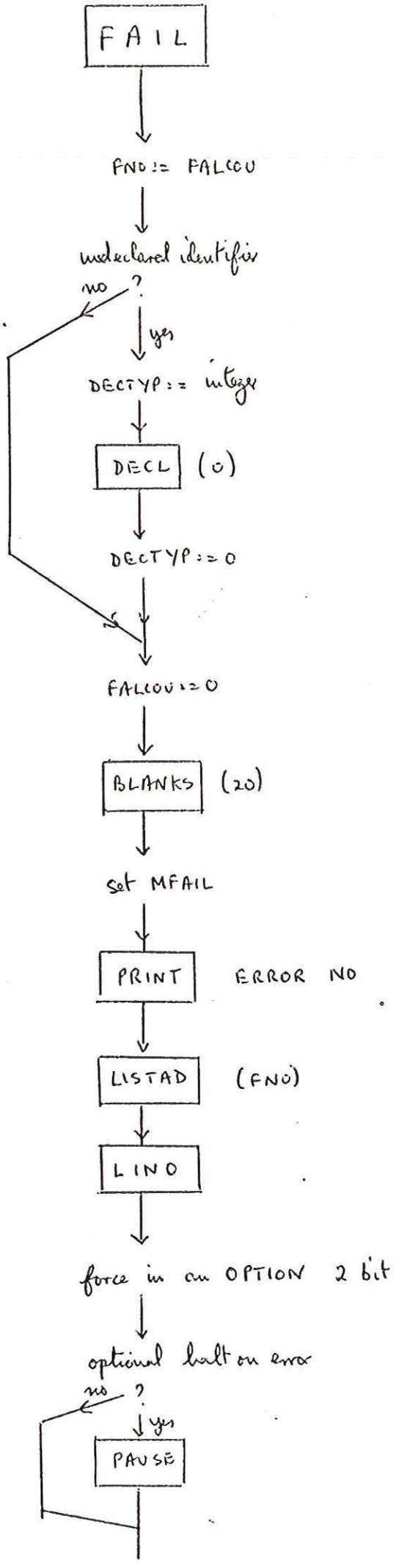
PROCED

COLON



CALLED FROM

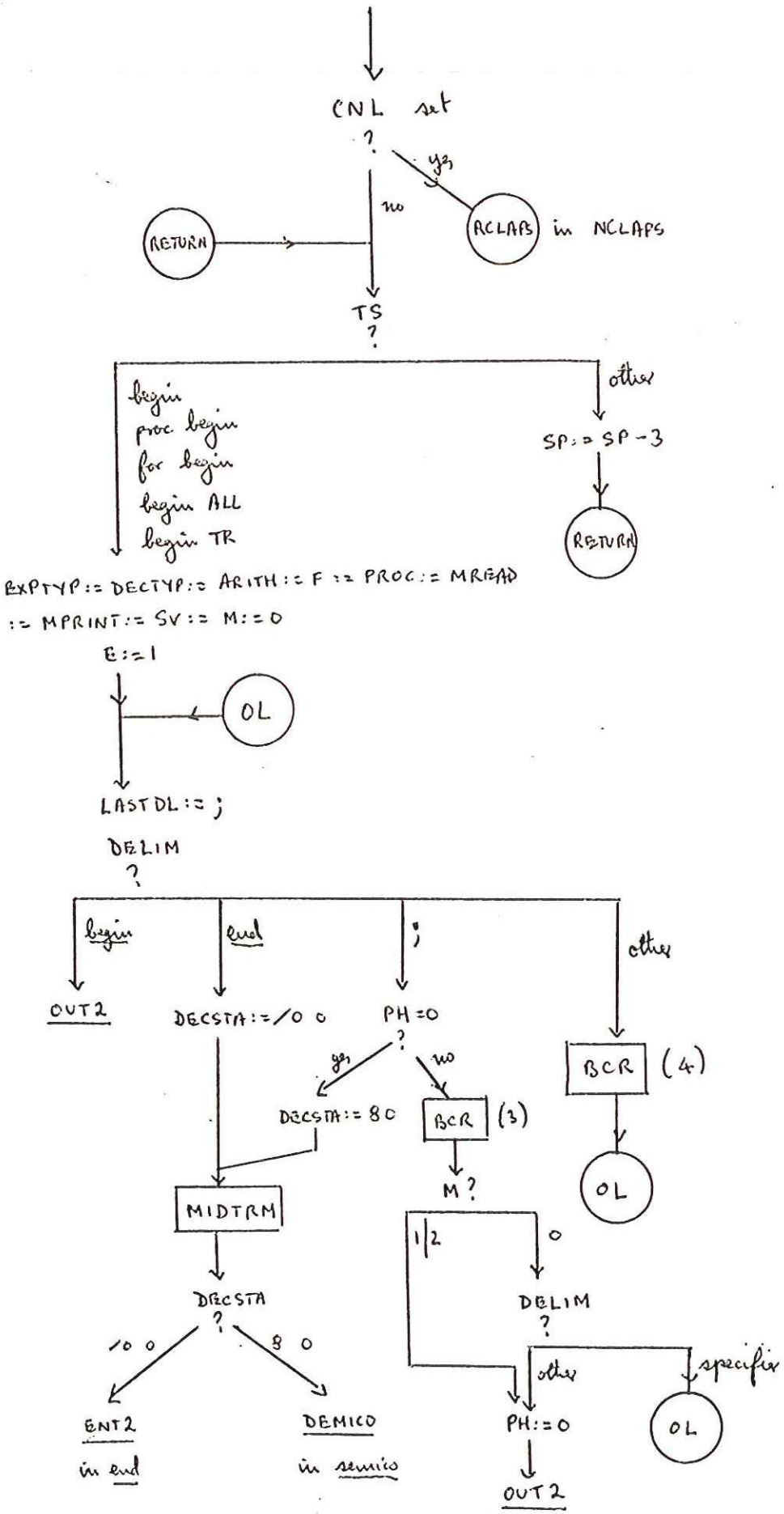
FAIL



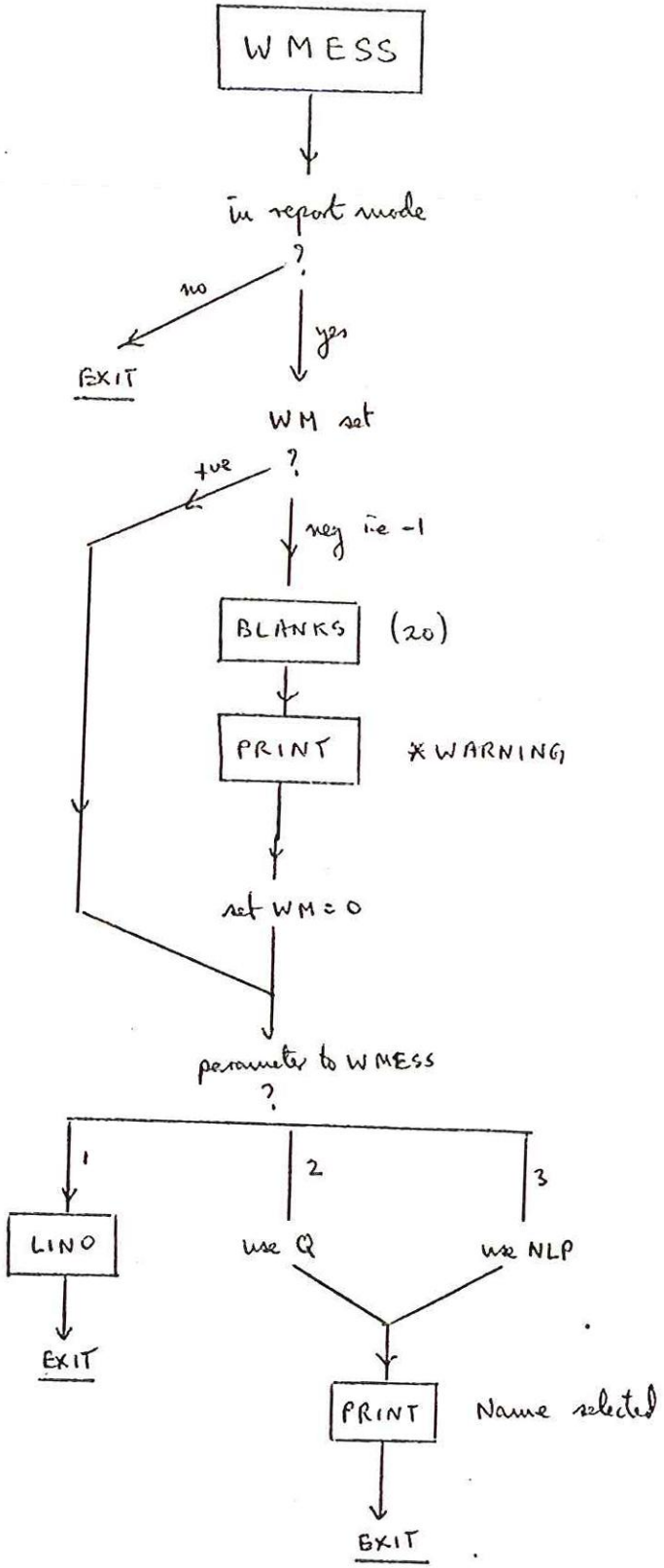
debugging facility

FAIL continued

for multiple fails during NCLAPS



MIDTRM is an entry in STATRM

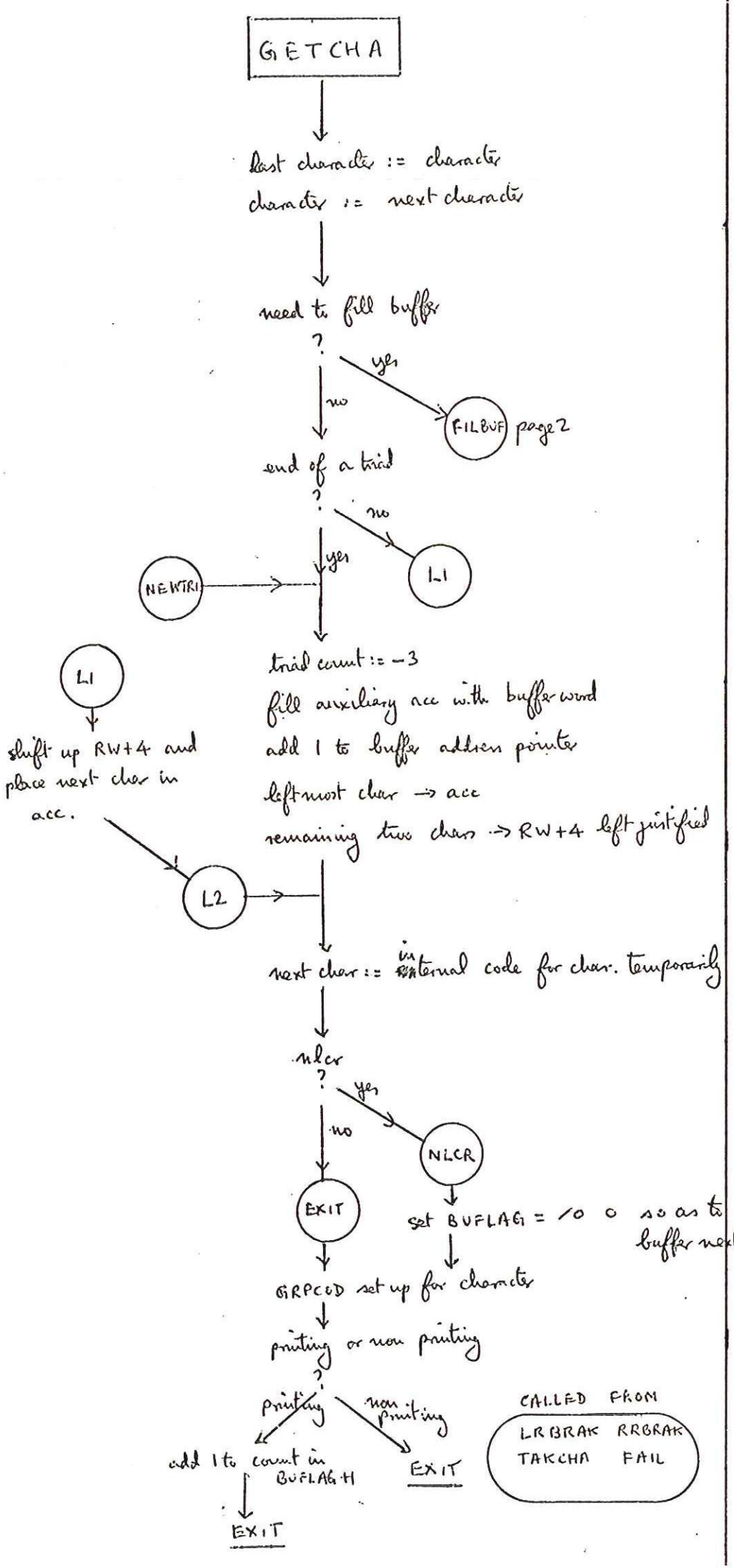


CALLED FROM
END FCLAPS
NCLAPS FAIL

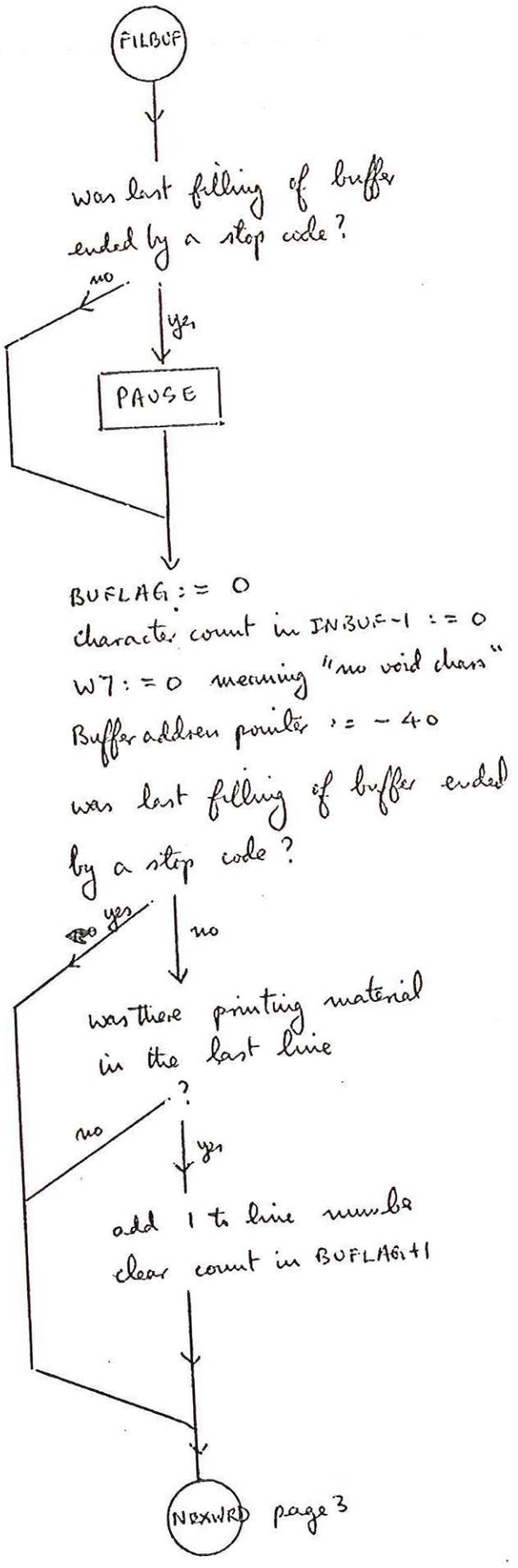
gives warning messages when in report mode

WM used to control suppression of the word *WARNING for a list of variables

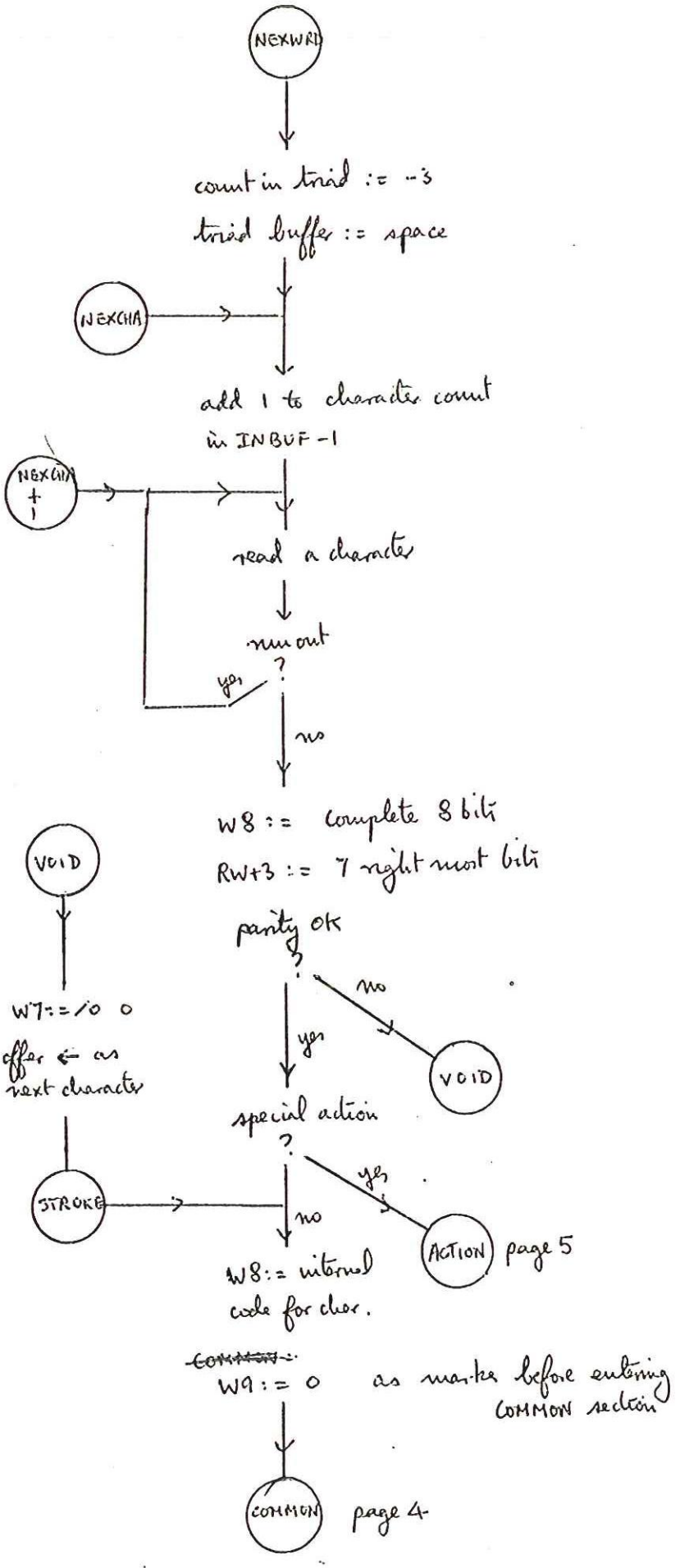
Supply next character



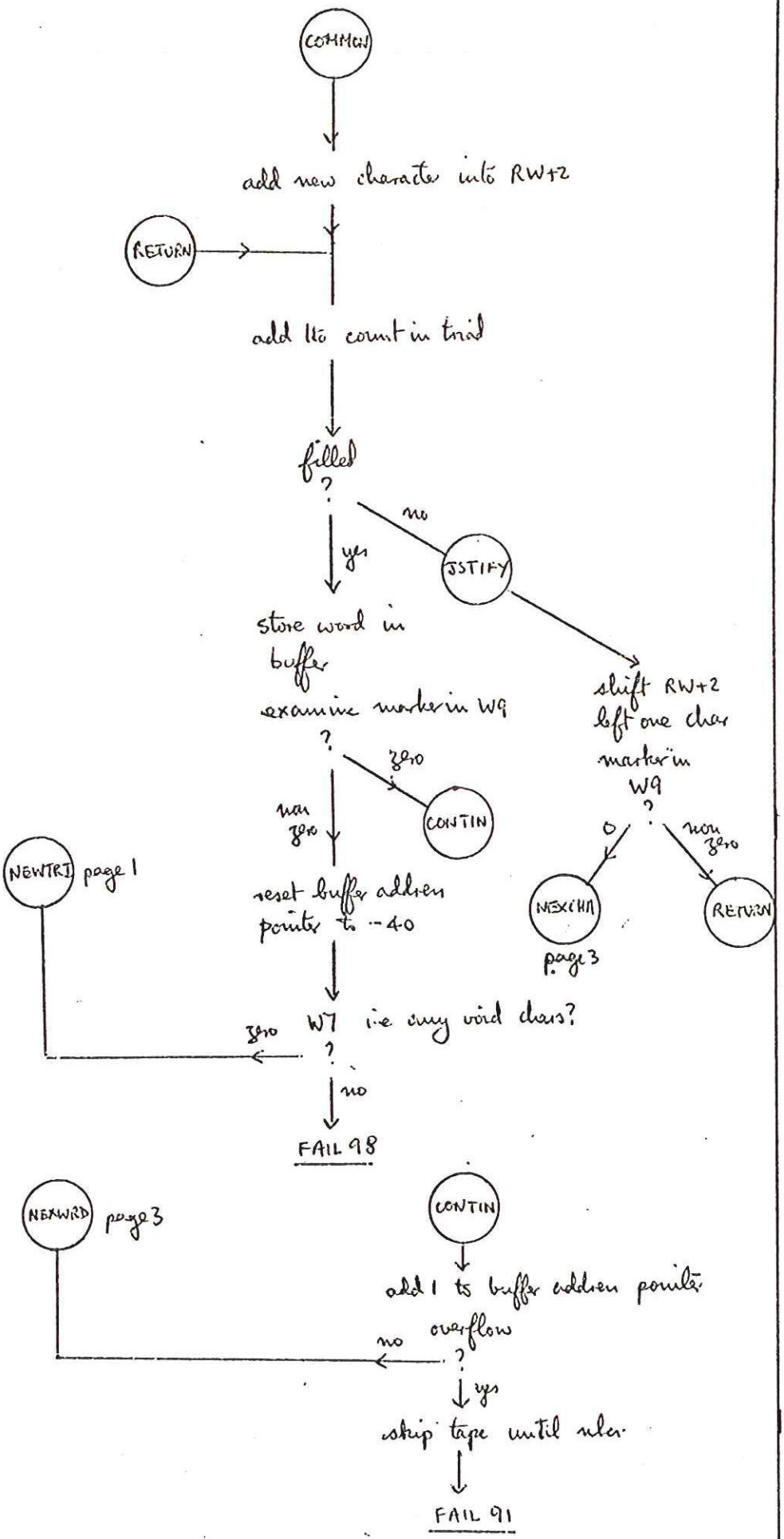
GETCHA


FILBUF
was last filling of buffer ended by a stop code?
no
yes
PAUSE
BUFLAG := 0
character count in INBUF-1 := 0
W7 := 0 meaning "no void chars"
Buffer address pointer := -40
was last filling of buffer ended by a stop code?
yes
no
was there printing material in the last line ?
no
yes
add 1 to line number
clear count in BUFLAG+1
NEXWRD
page 3

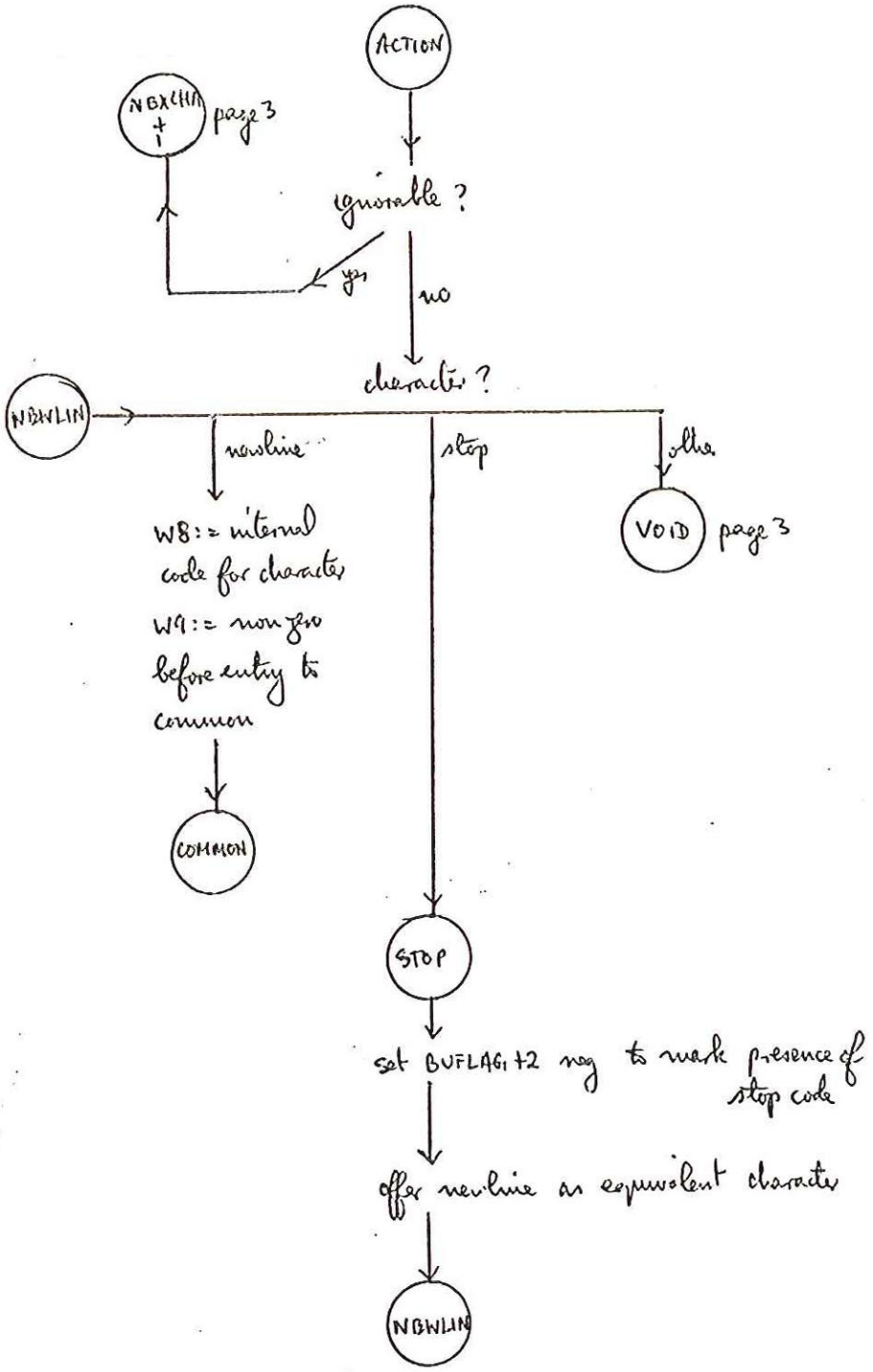
GET CHA continued

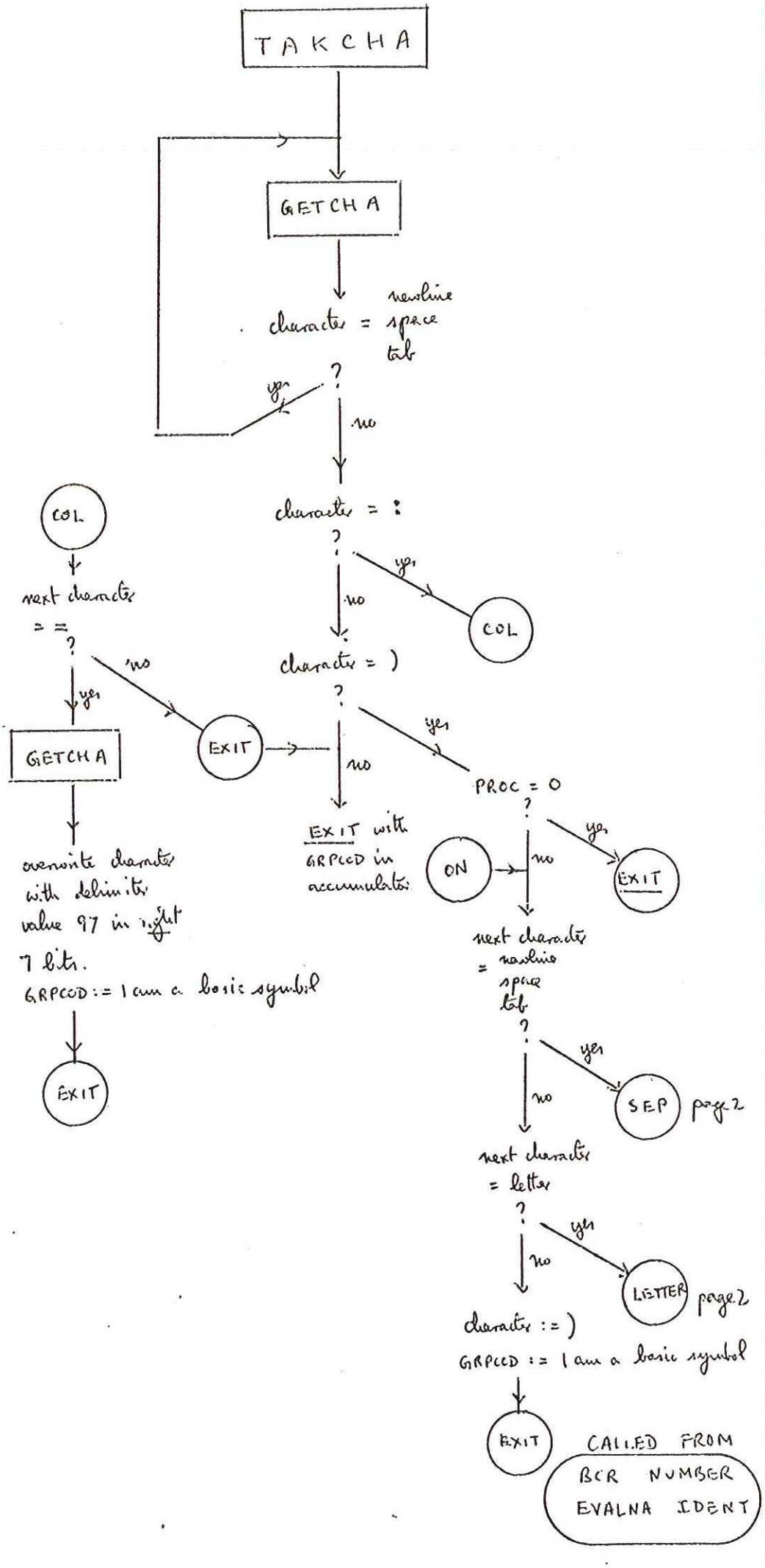


GETCHA continued

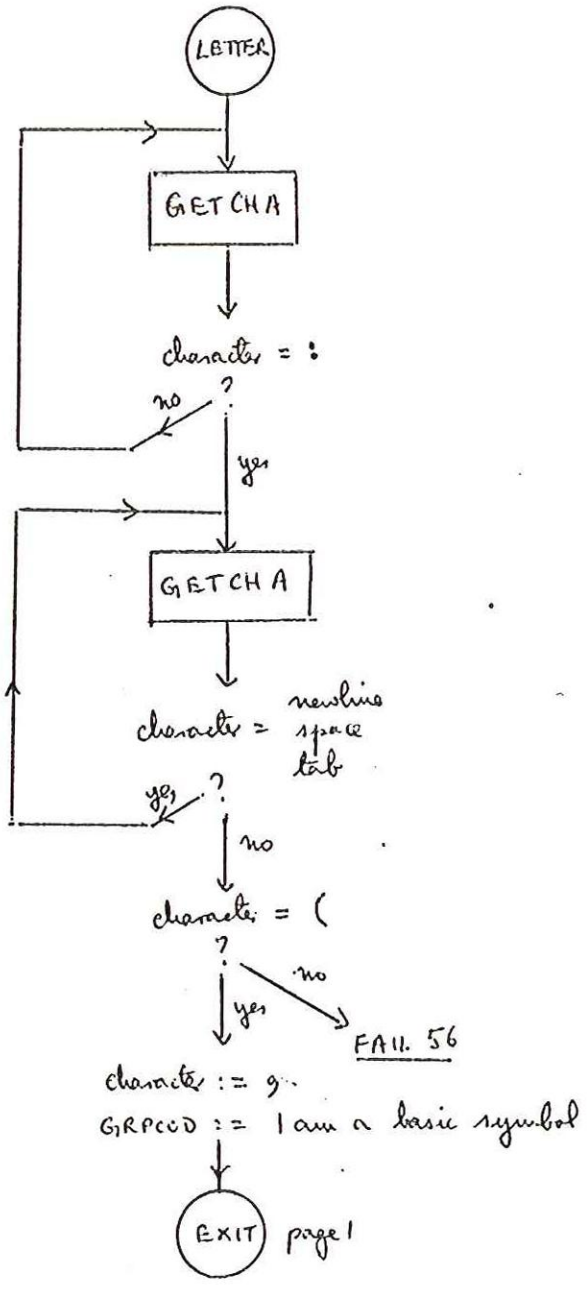


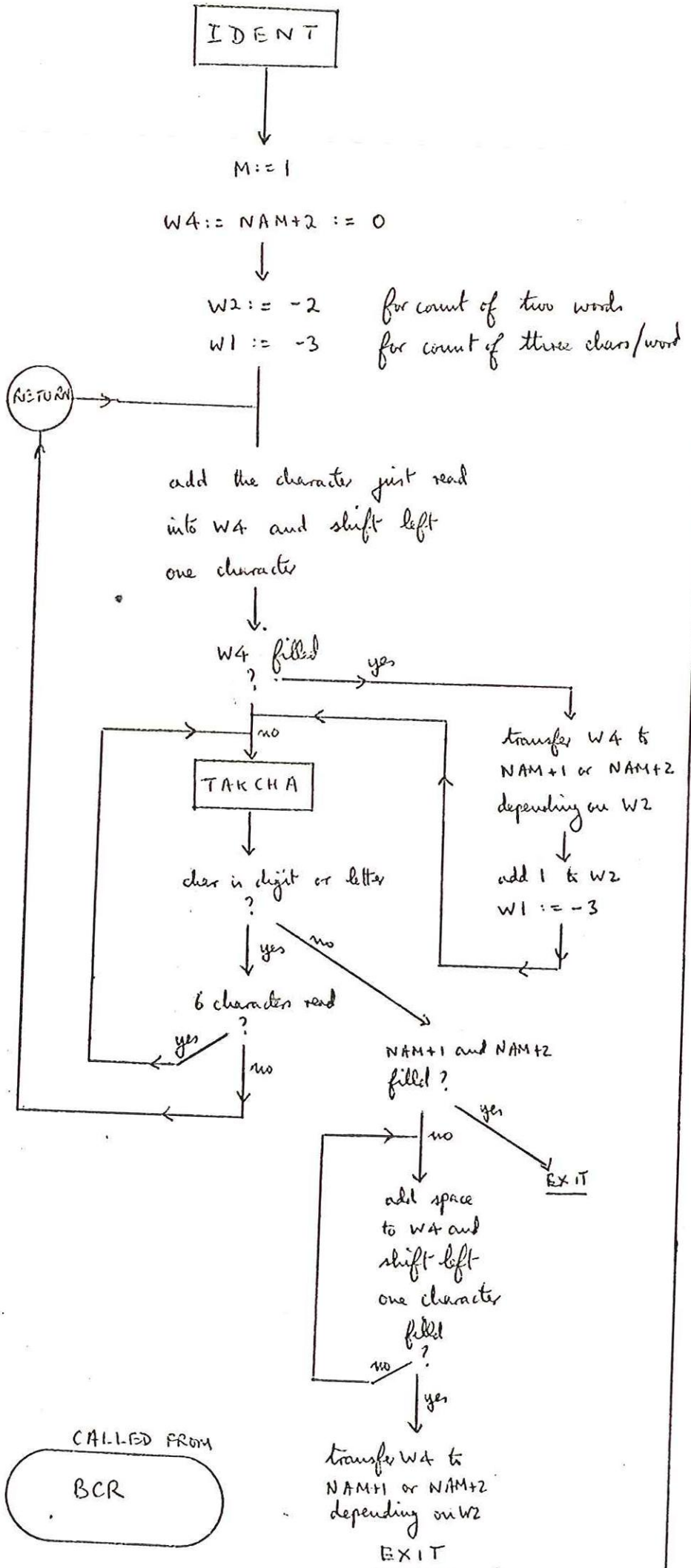
GETCHA continued



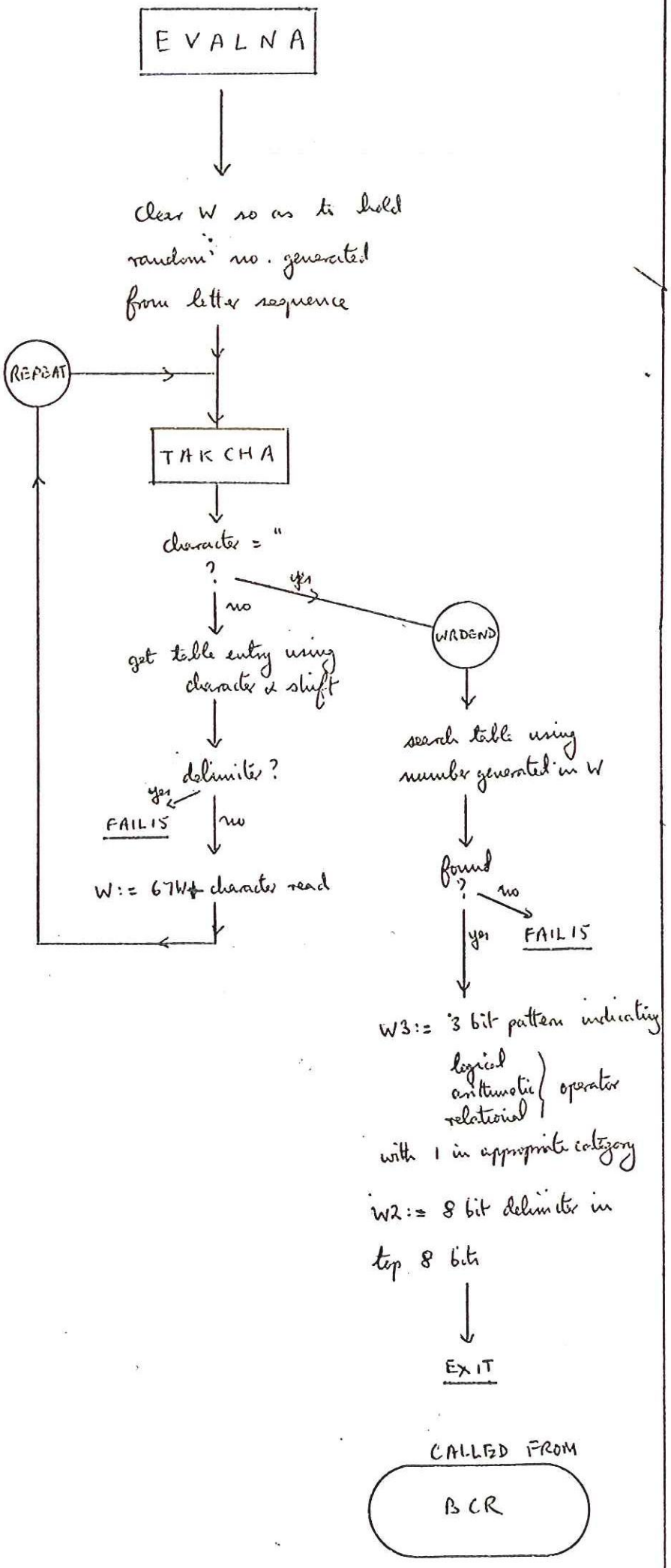

TAKCHA
GETCHA
character = newline space tab
?
yes
no
character = :
?
yes
COL
no
character =)
?
yes
PROC = 0
?
yes
EXIT
no
ON
no
next character = newline space tab
?
yes
SEP
page 2
no
next character = letter
?
yes
LETTER
page 2
no
character :=)
GRPCOD := I am a basic symbol
EXIT
COL
next character = =
?
yes
no
EXIT
GETCHA
overwrite character with delimiter value 97 in right 7 bits.
GRPCOD := I am a basic symbol
EXIT with GRPCOD in accumulator
CALLED FROM
BCR NUMBER
EVALNA IDENT

TAKCHA continued



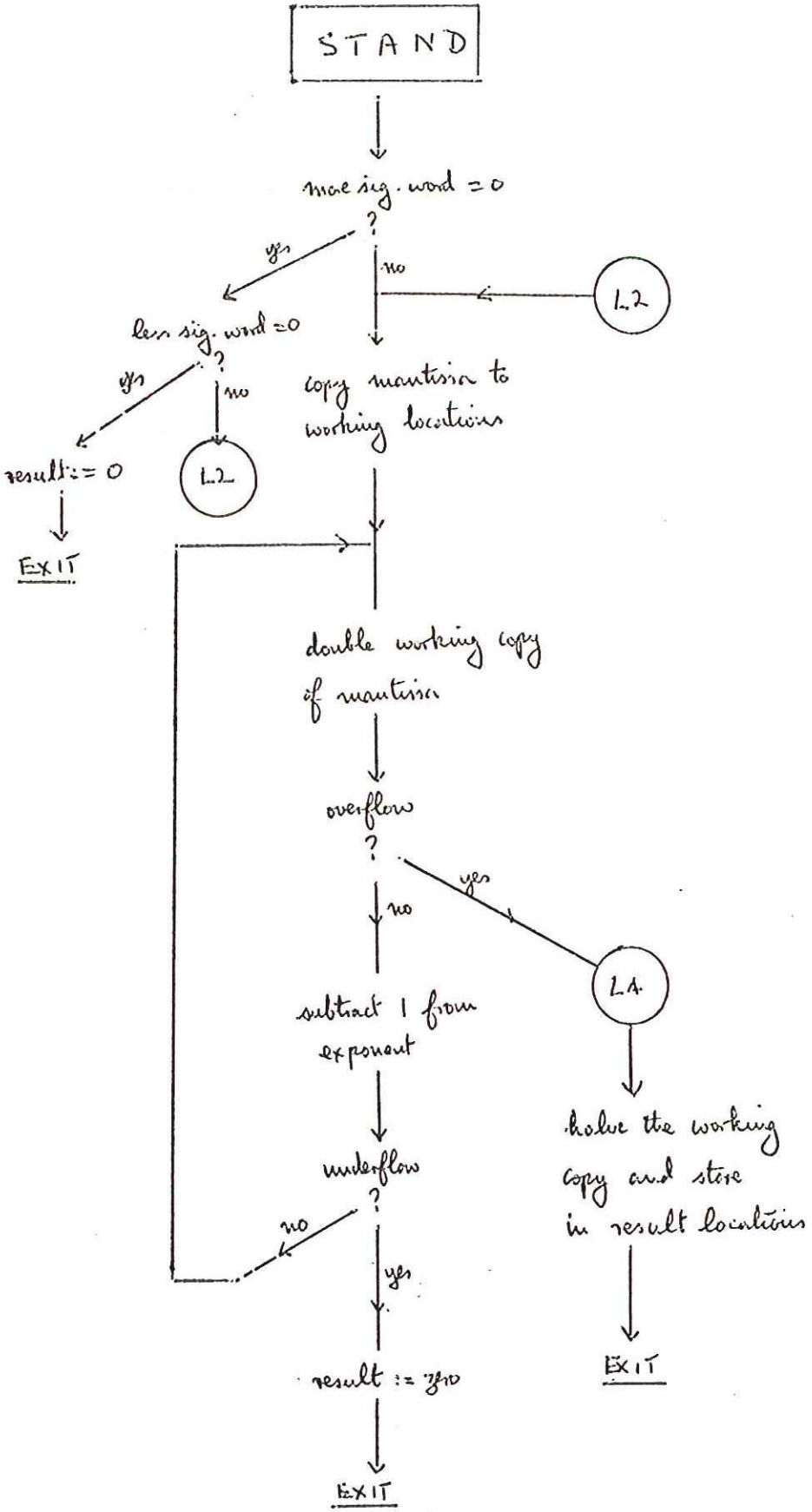


select up to first 6 chars of identifier



convert "mnemonic" word to delimiter value

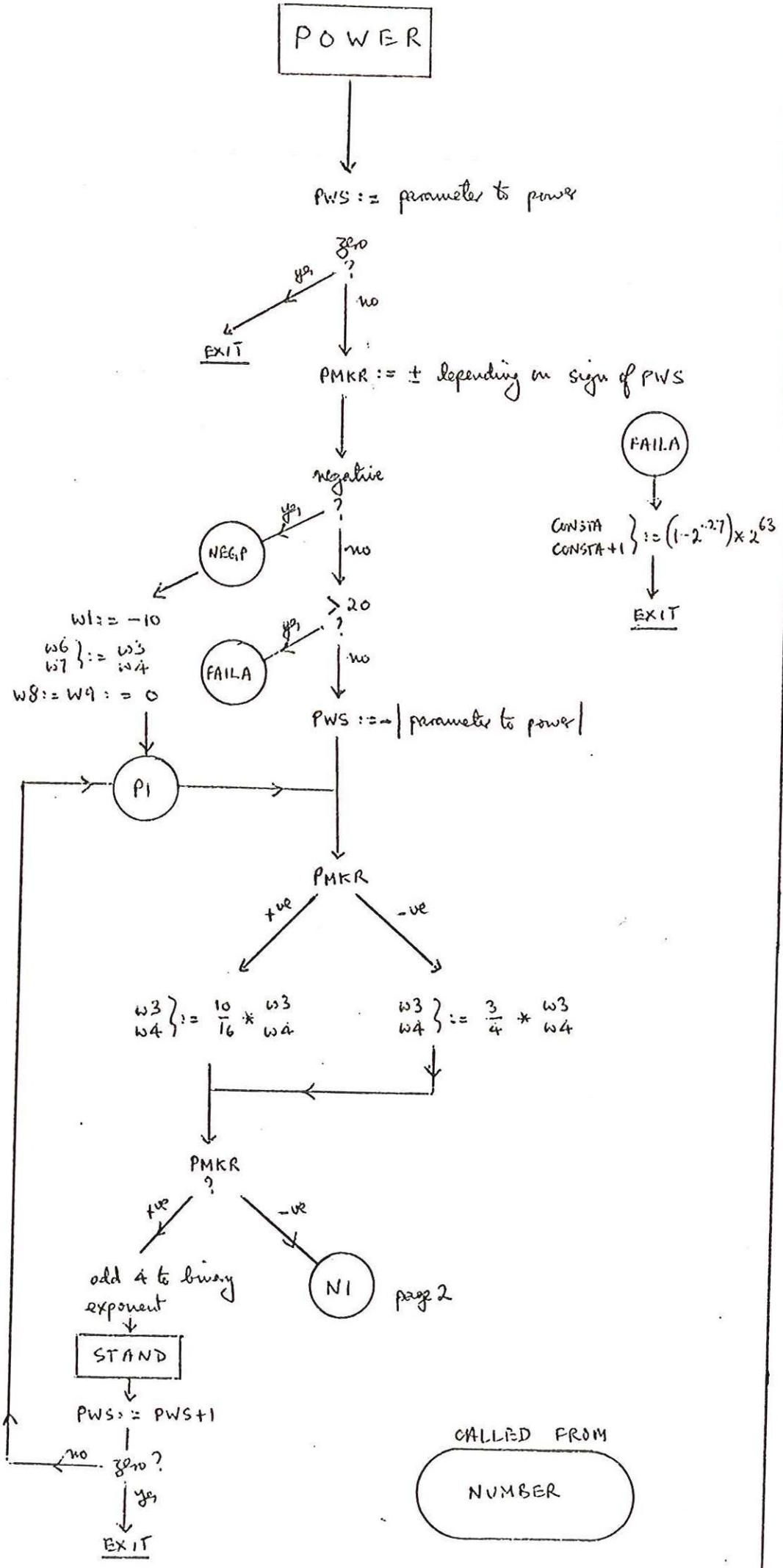
standardise contents of W3 W4 W5



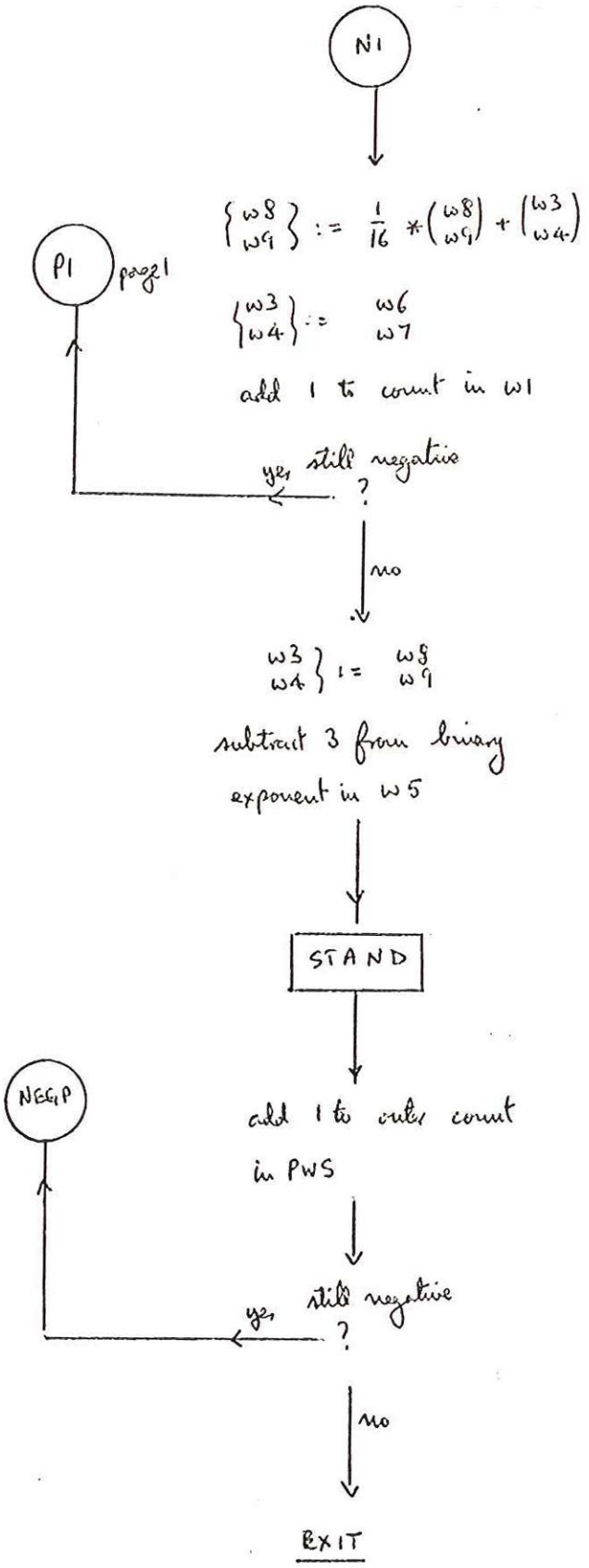
CALLED FROM

NUMBER

raise contents of w3 w4 w5 to power of ten given in Acc.

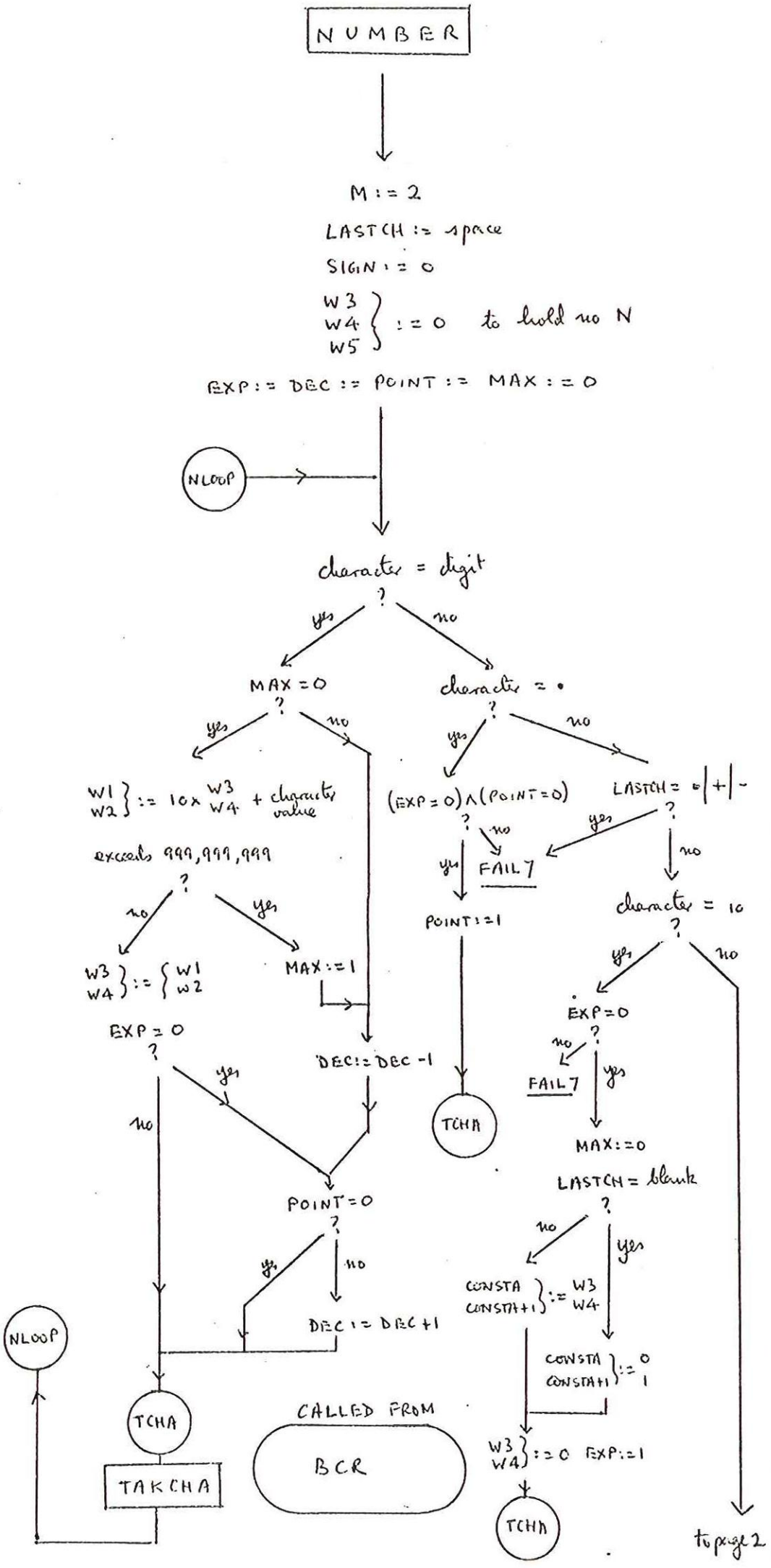


POWER continued

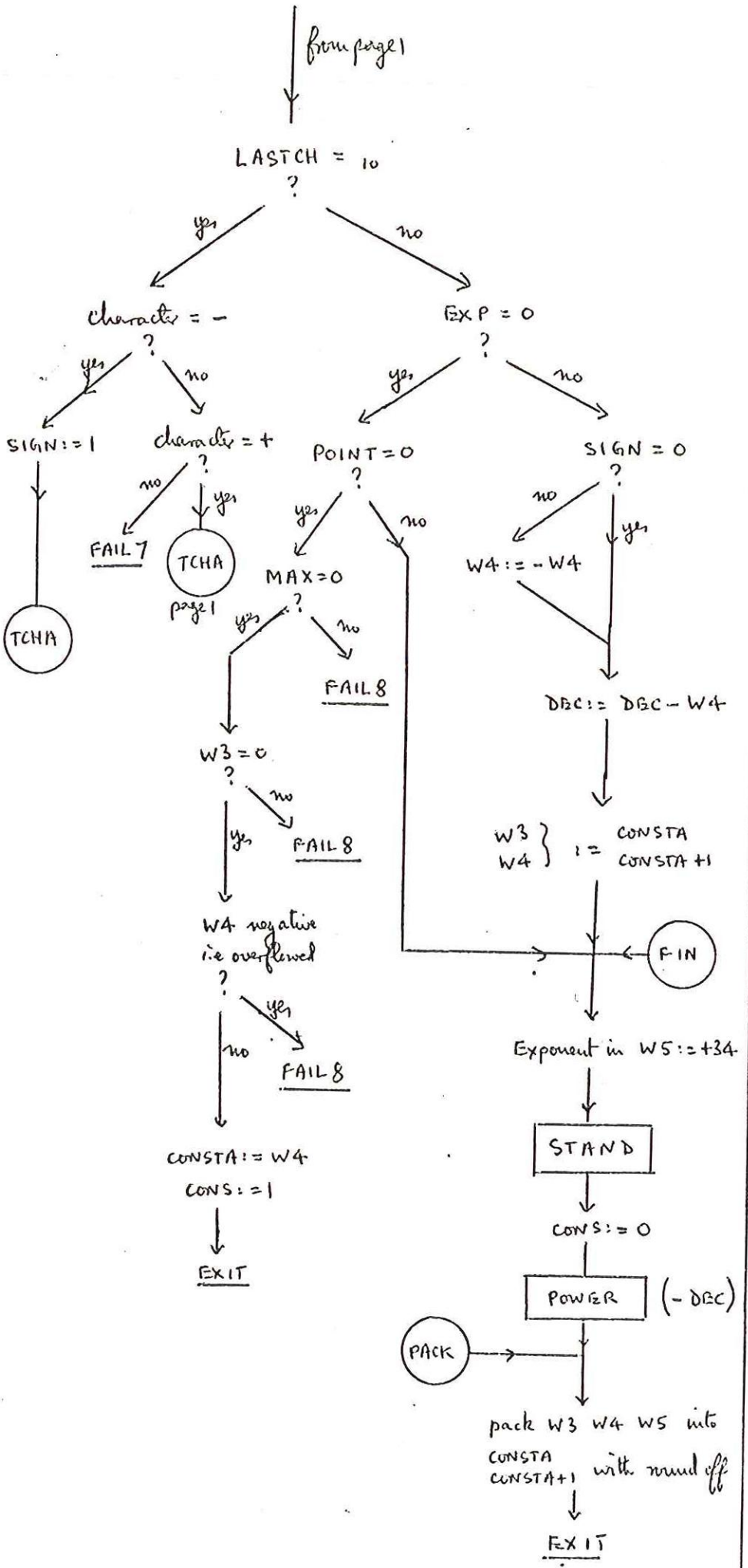


This is a programmed multiplication by the constant 0.0001100111.... to 40 binary places. This is $\frac{1}{10}$.

convert input no. to floating binary.

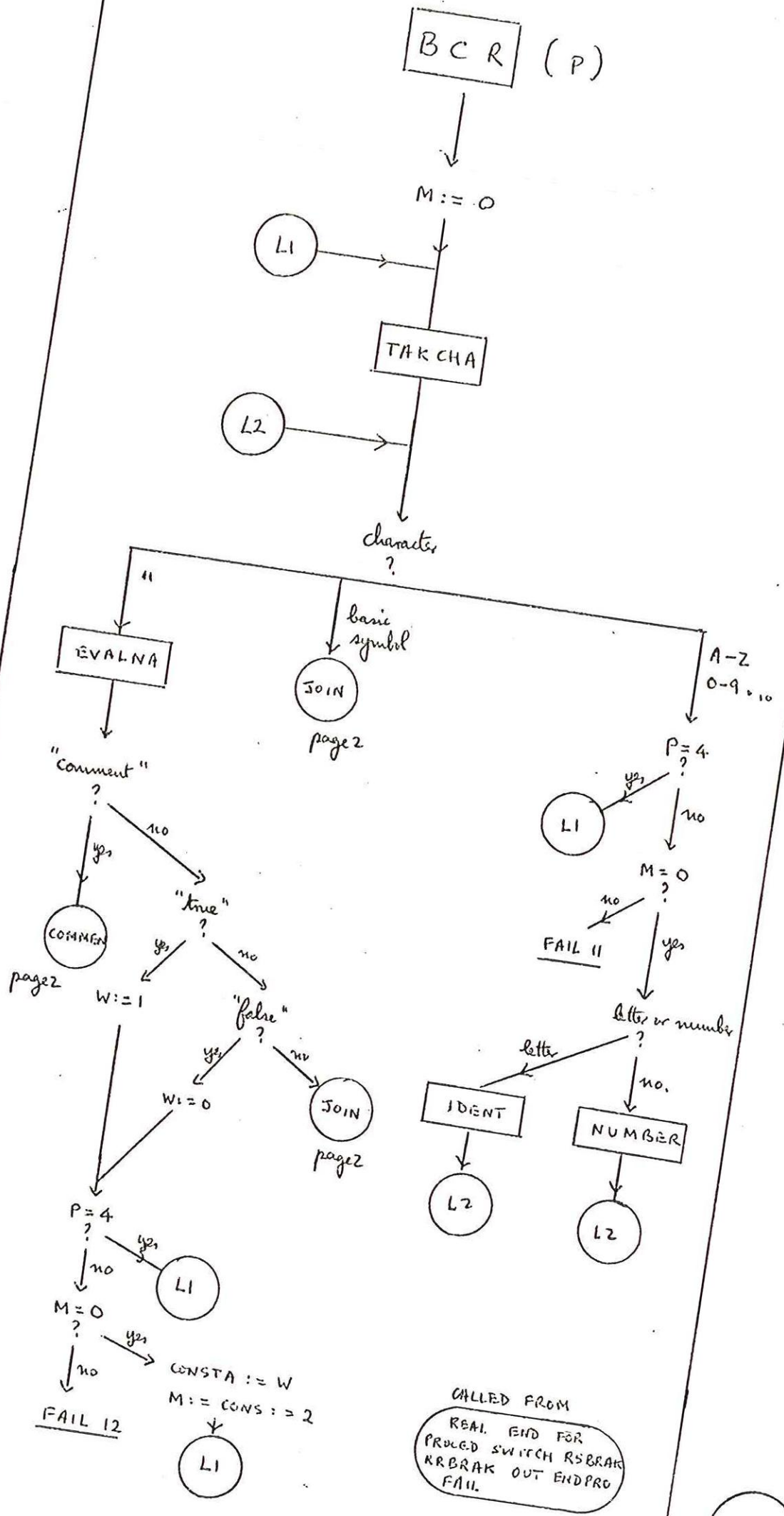


NUMBER continued

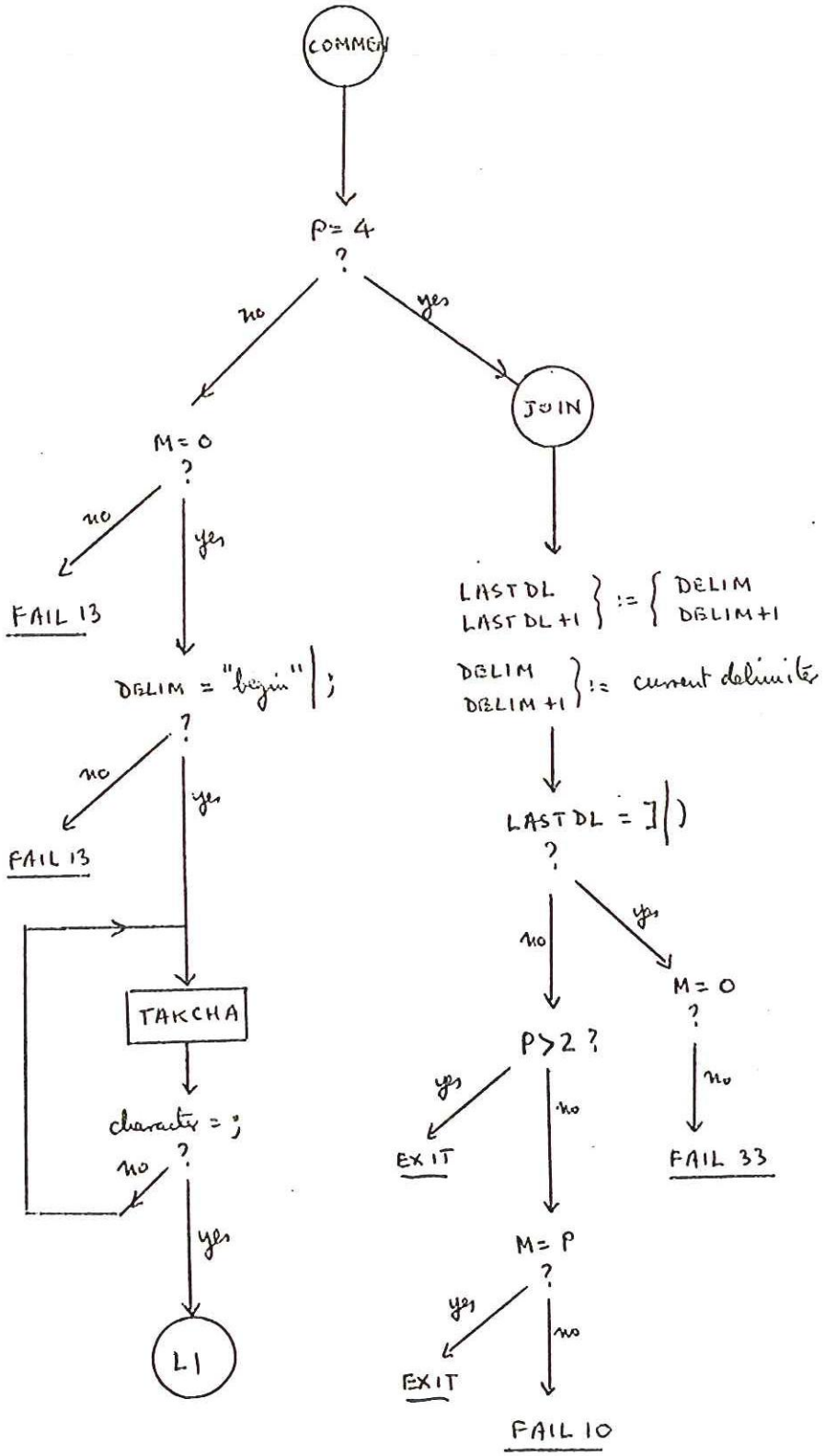


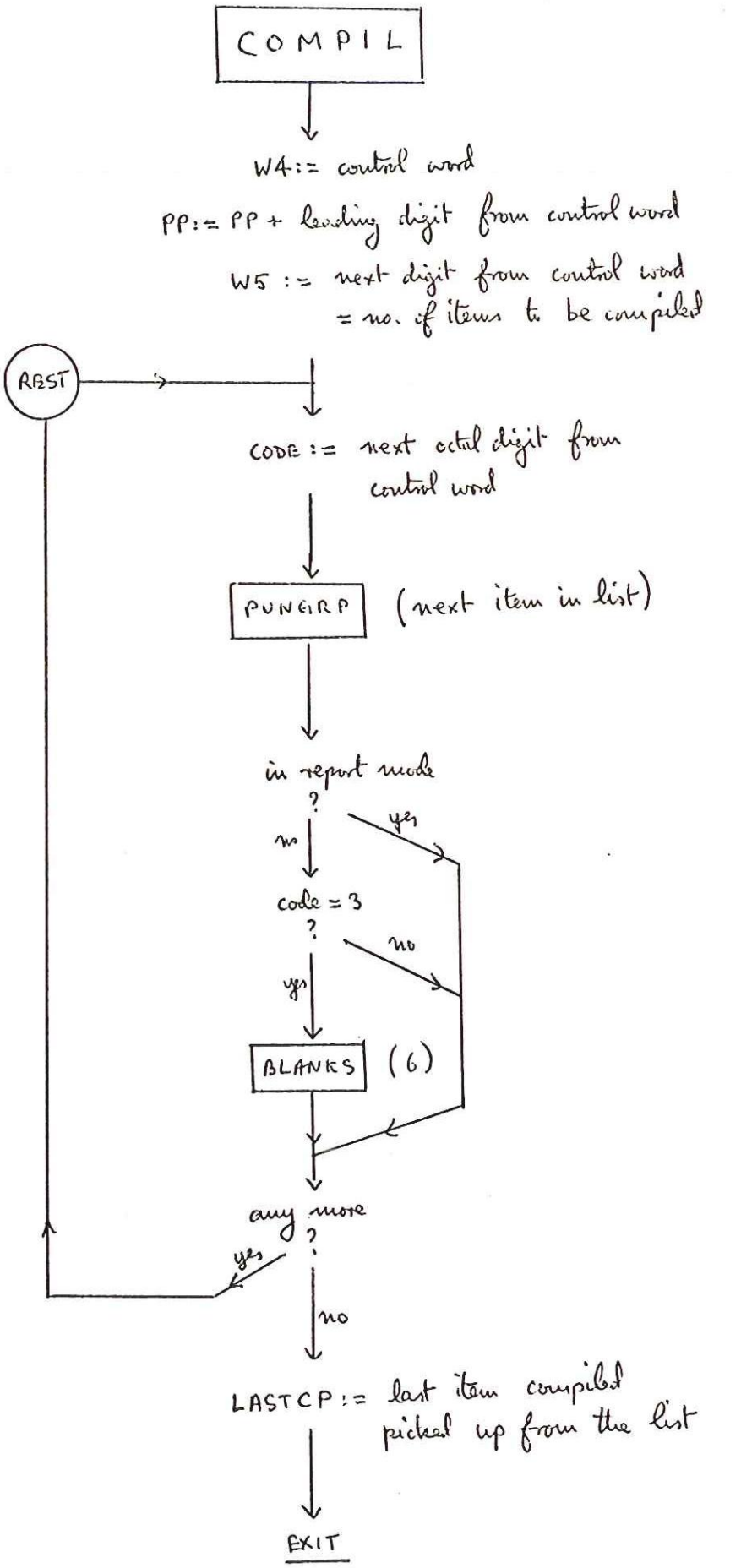
We are dealing with the exponent hence it is single length in W4

if real no. too big replace by $(1-2^{-27}) \times 2^{63}$


BCR (P)
M := 0
L1
TAKCHA
L2
character ?
"
basic symbol
A-Z
0-9 . 10
EVALNA
JOIN
page 2
"comment" ?
yes
no
COMMEN
page 2
"true" ?
yes
no
W := 1
"false" ?
yes
no
W := 0
JOIN
page 2
P = 4 ?
yes
no
L1
M = 0 ?
no
FAIL 12
yes
CONSTA := W
M := CONS := 2
L1
P = 4 ?
yes
no
L1
M = 0 ?
no
FAIL 11
yes
letter or number ?
letter
no.
IDENT
L2
NUMBER
L2
CALLED FROM
REAL END FOR PROCED SWITCH RSBRAK KRBRAK OUT ENDPRO FAIL.

BCR continued





<control word>
::= $O_1 O_2 O_3 O_4 O_5 O_6$

$PP := PP + O_1$

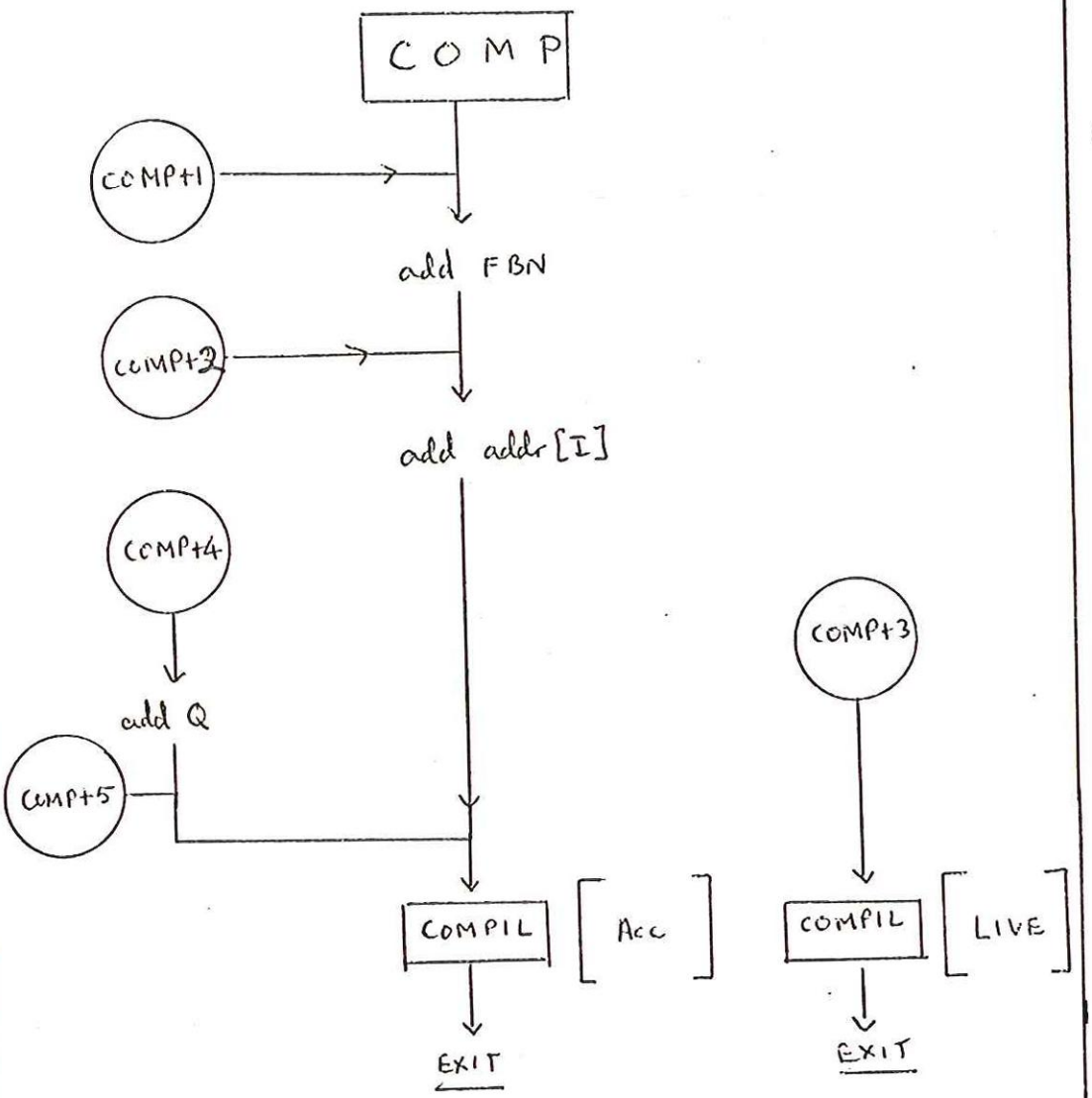
O_2 = no of words to compile

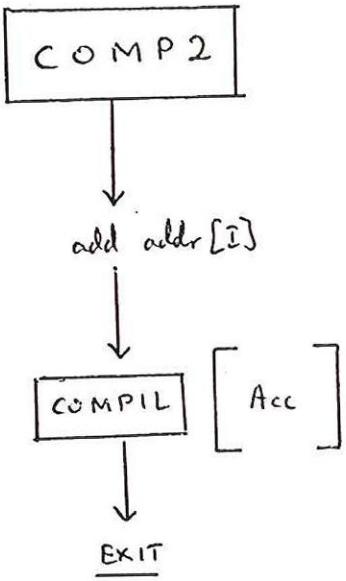
$O_3 - O_6$ = codes

CALLED FROM

ALMOST EVERYWHERE

Compiles with SIR loader code 1





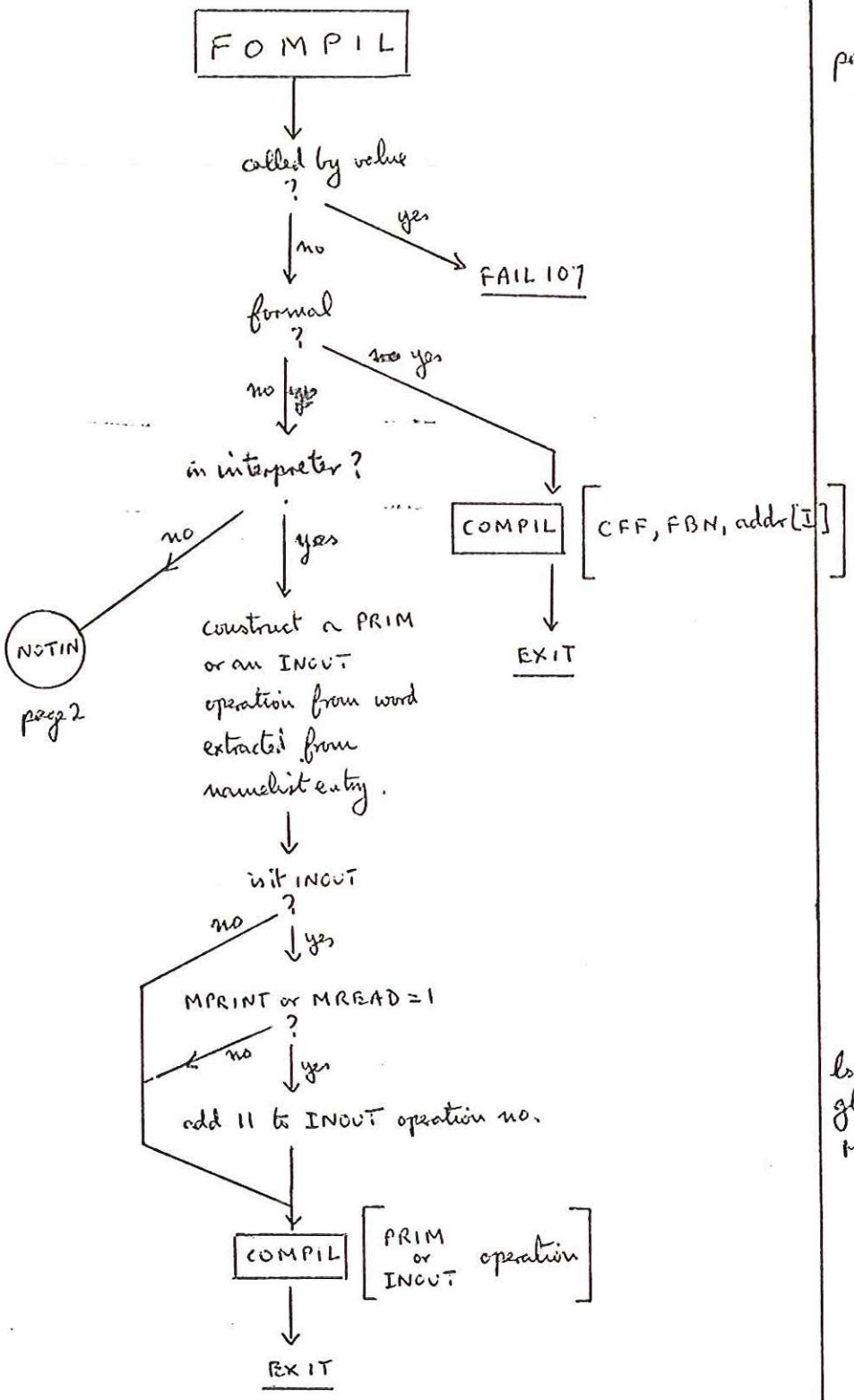
This is with SIR loader code 2

COMP2 CALLED FROM

FOMPIL PRAMCH THKID

COMP CALLED FROM

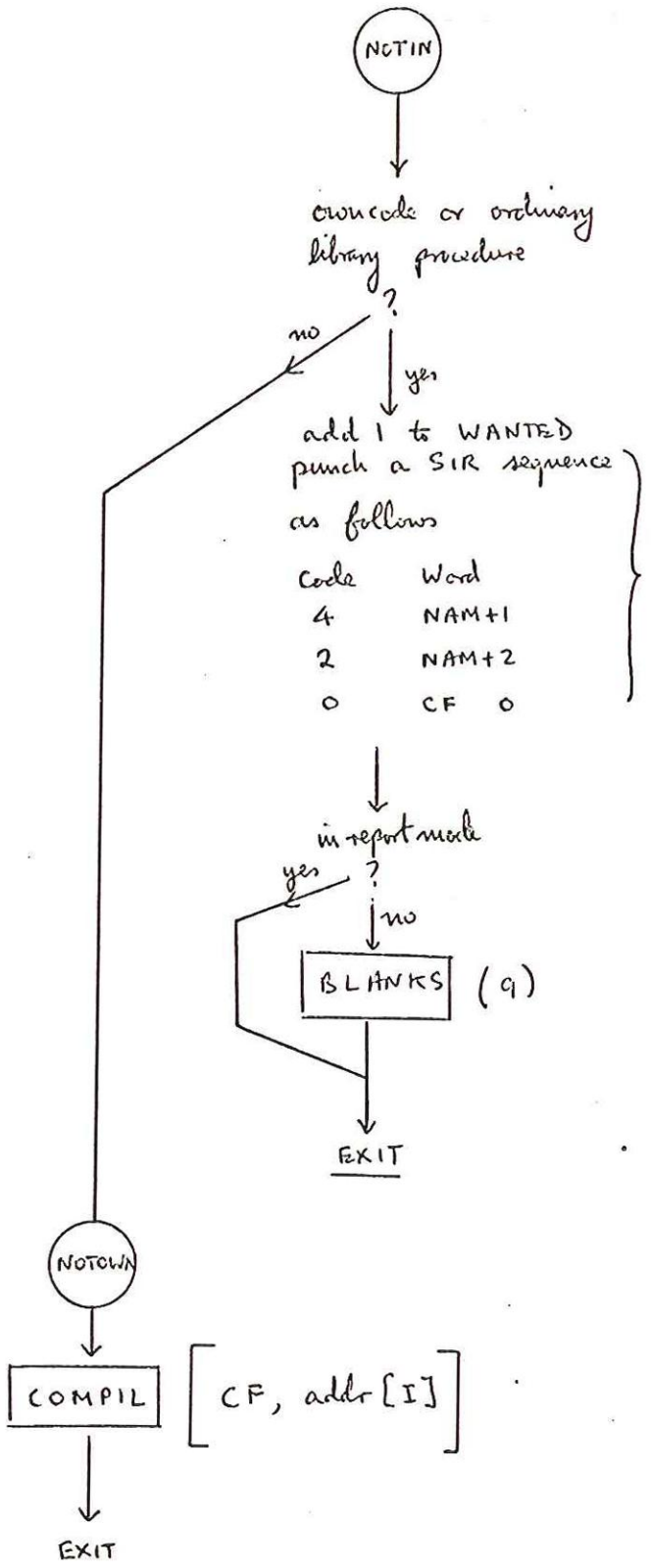
ALMOST EVERYWHERE



local not global if MREAD or MPRINT set

CALLED FROM
RRBRAK FOMCOM INOUT

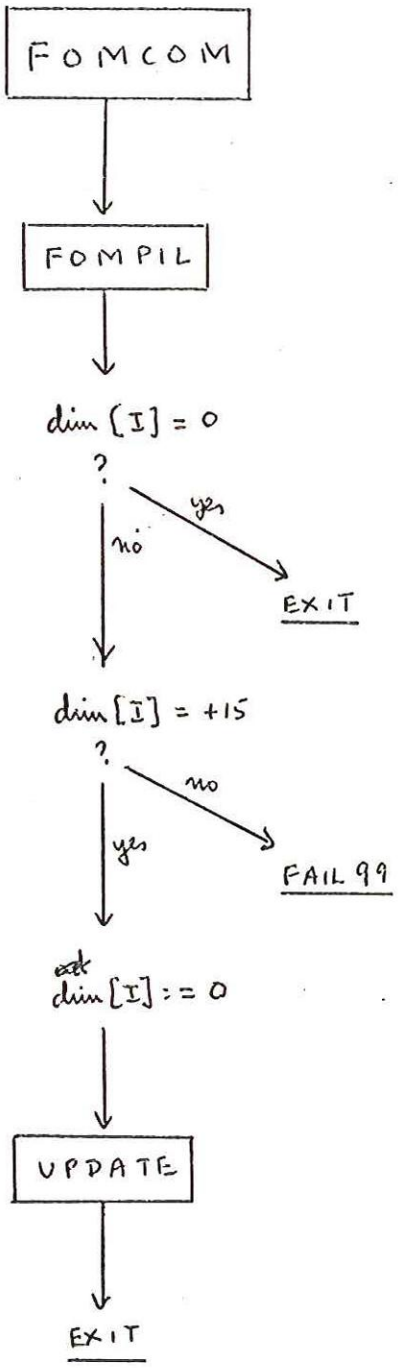
FOMPIL continued



this is suppressed inside PUNCRP in report mode

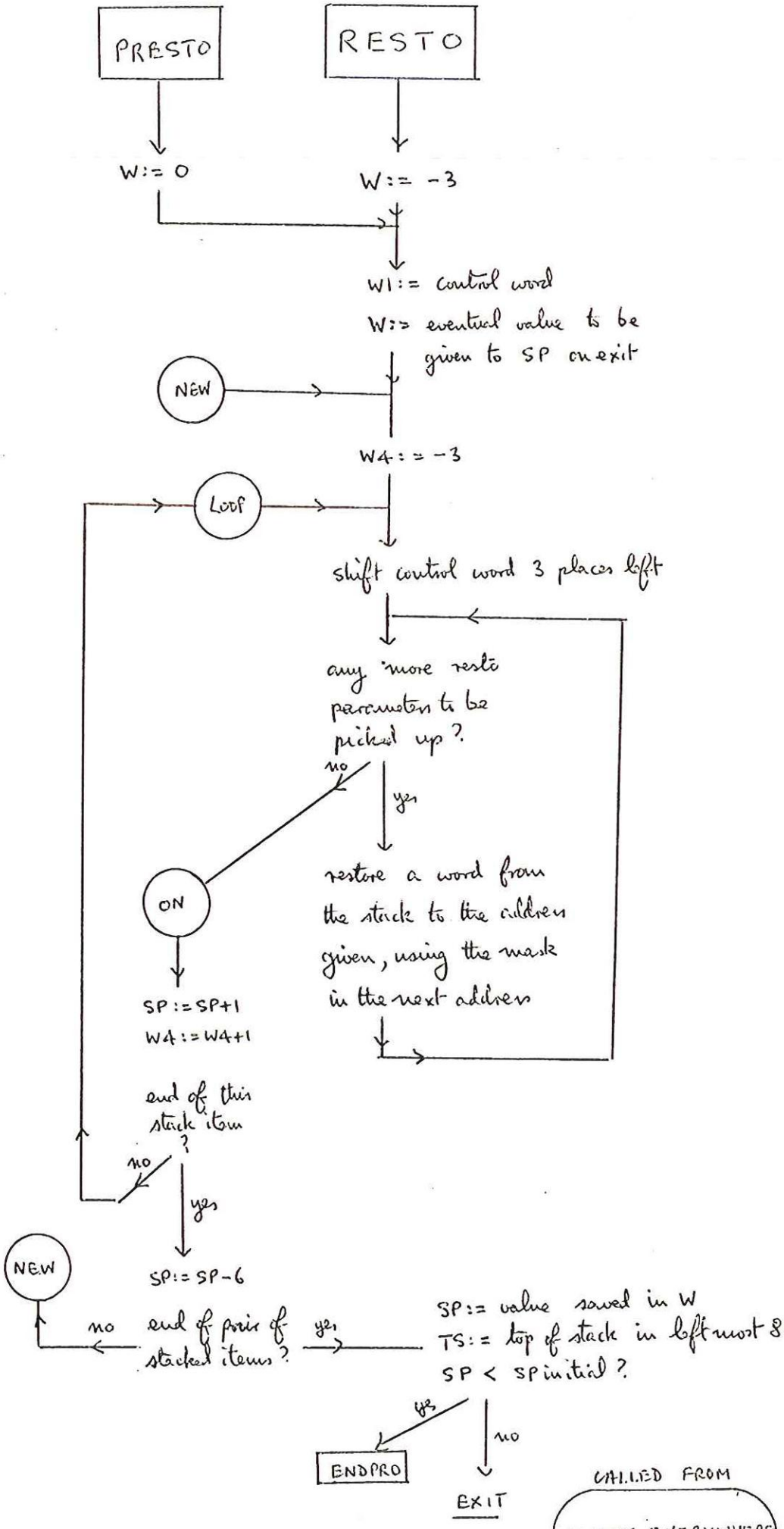
This is the normal library call which is dealt with at load time

This is the ordinary case. Word is compiled with SIR code 2

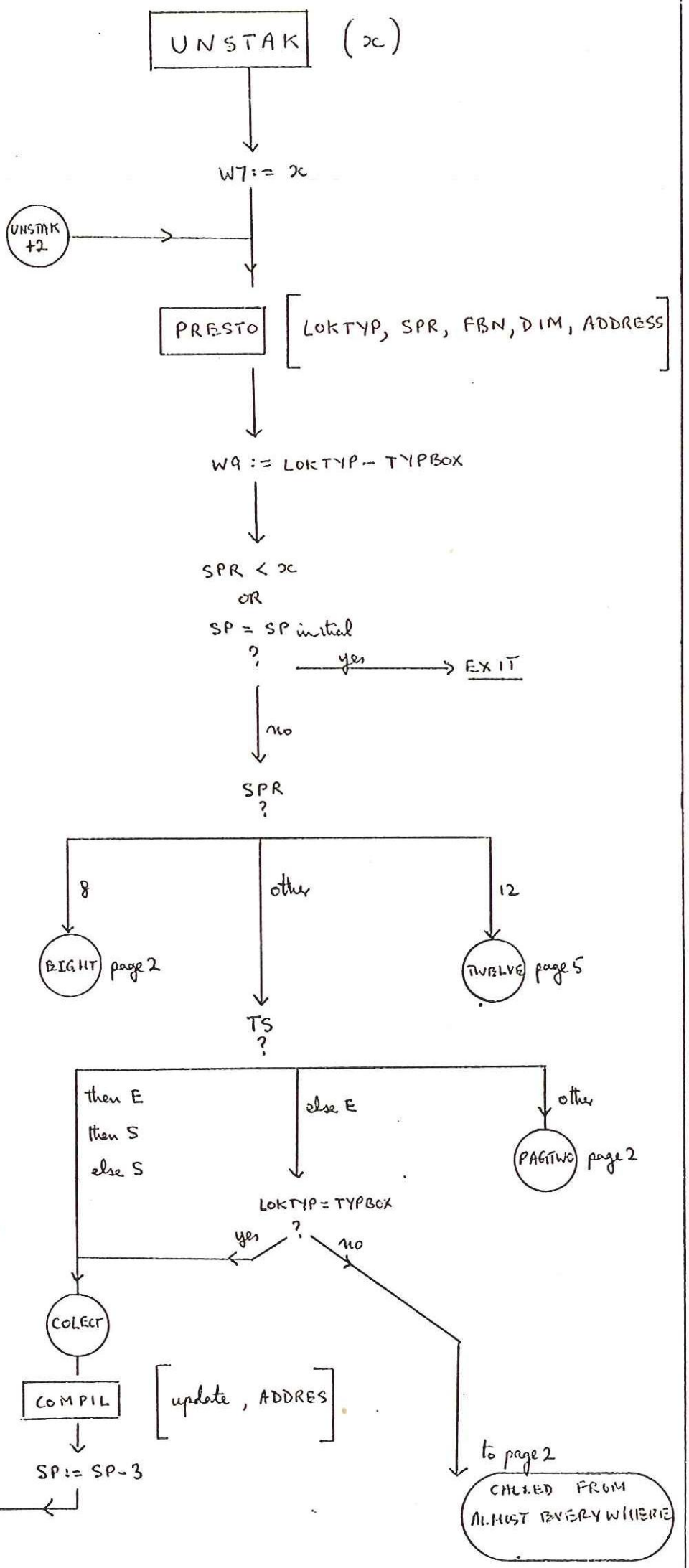


CALLED FROM

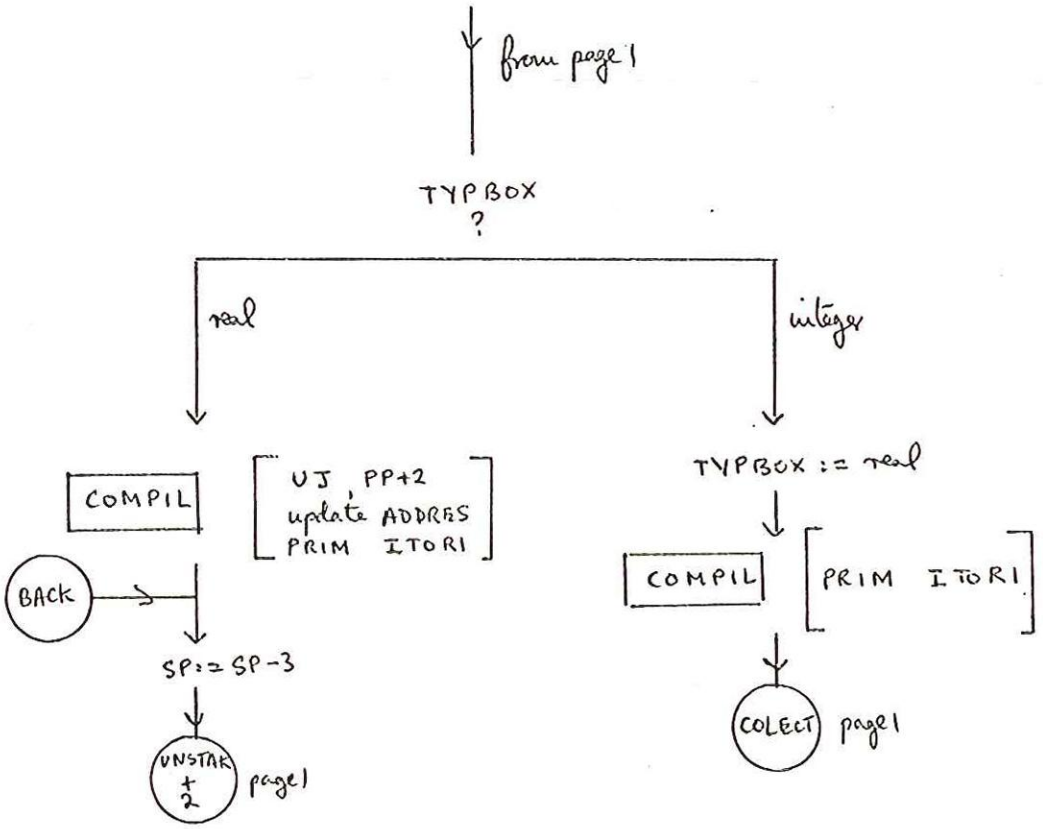
PRAMCH TAKID
ENDSTA

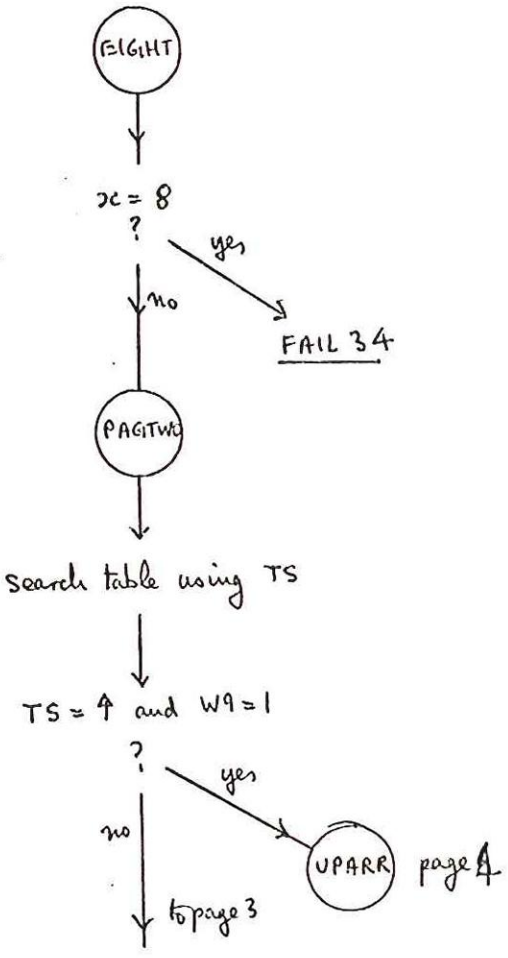


ENDPRO is reached to stop translation

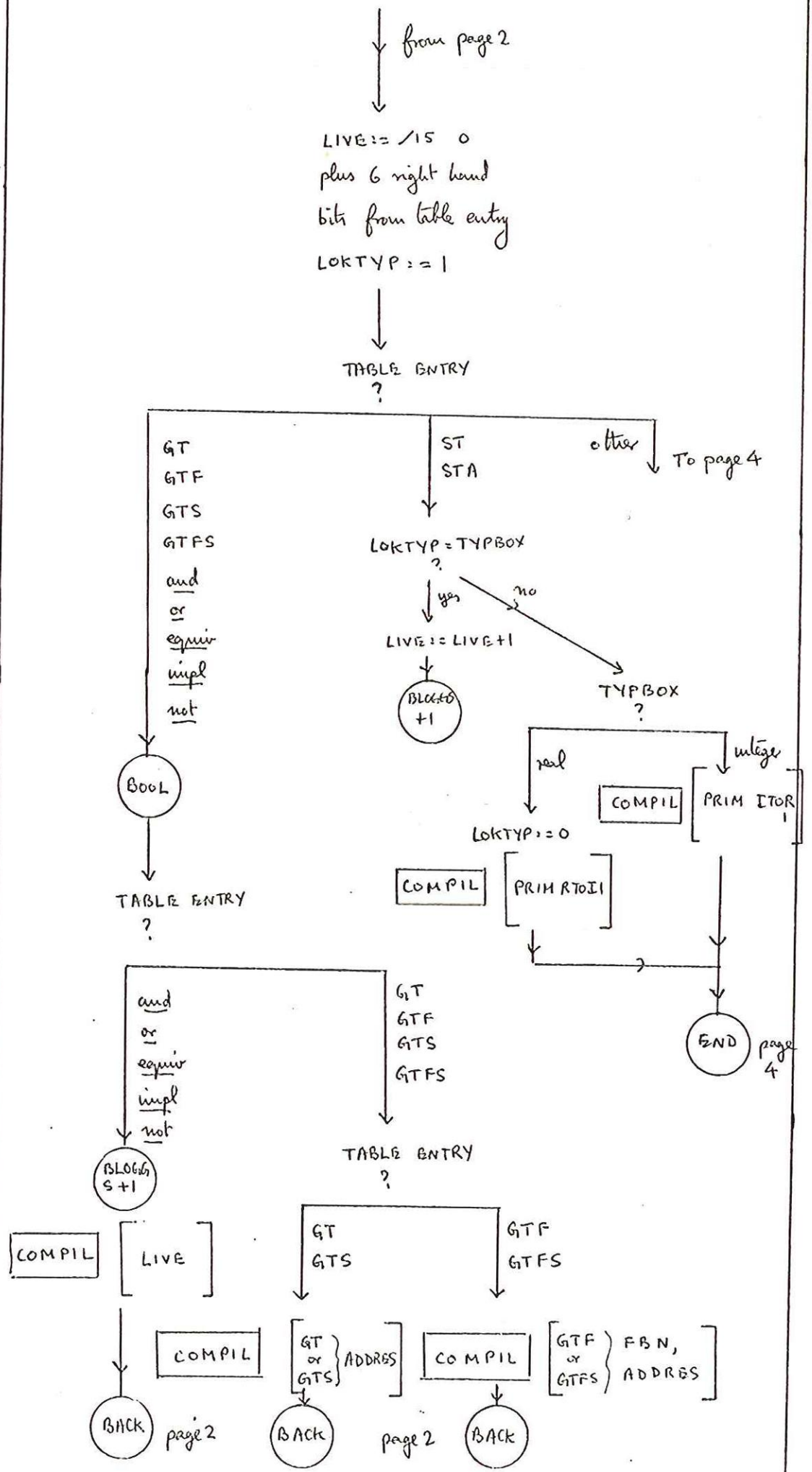

UNSTAK (x)
W7 := x
UNSTAK +2
PRESTO
[LOKTYP, SPR, FBN, DIM, ADDRESS]
SPR is the stacked priority
W9 := LOKTYP -- TYPBOX
SPR < x
OR
SP = SP initial
?
yes
EXIT
no
SPR
?
8
other
12
EIGHT page 2
TWELVE page 5
TS
?
then E
then S
else S
else E
other
PAGTWO page 2
LOKTYP = TYPBOX
?
yes
no
UNSTAK +2
COLECT
COMPIL
[update, ADDRES]
SP := SP-3
to page 2
CALLED FROM ALMOST EVERYWHERE

UNSTAK continued

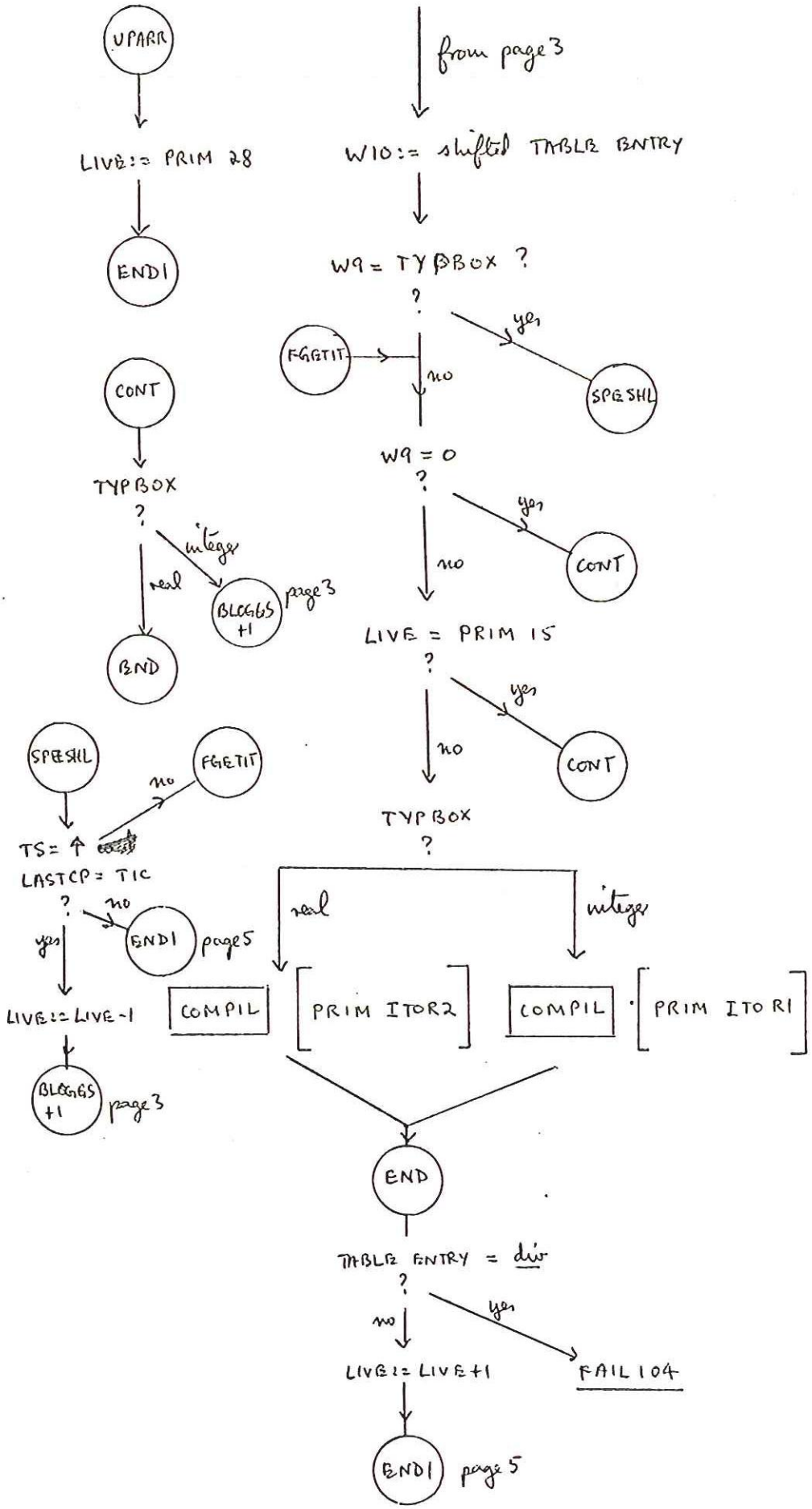




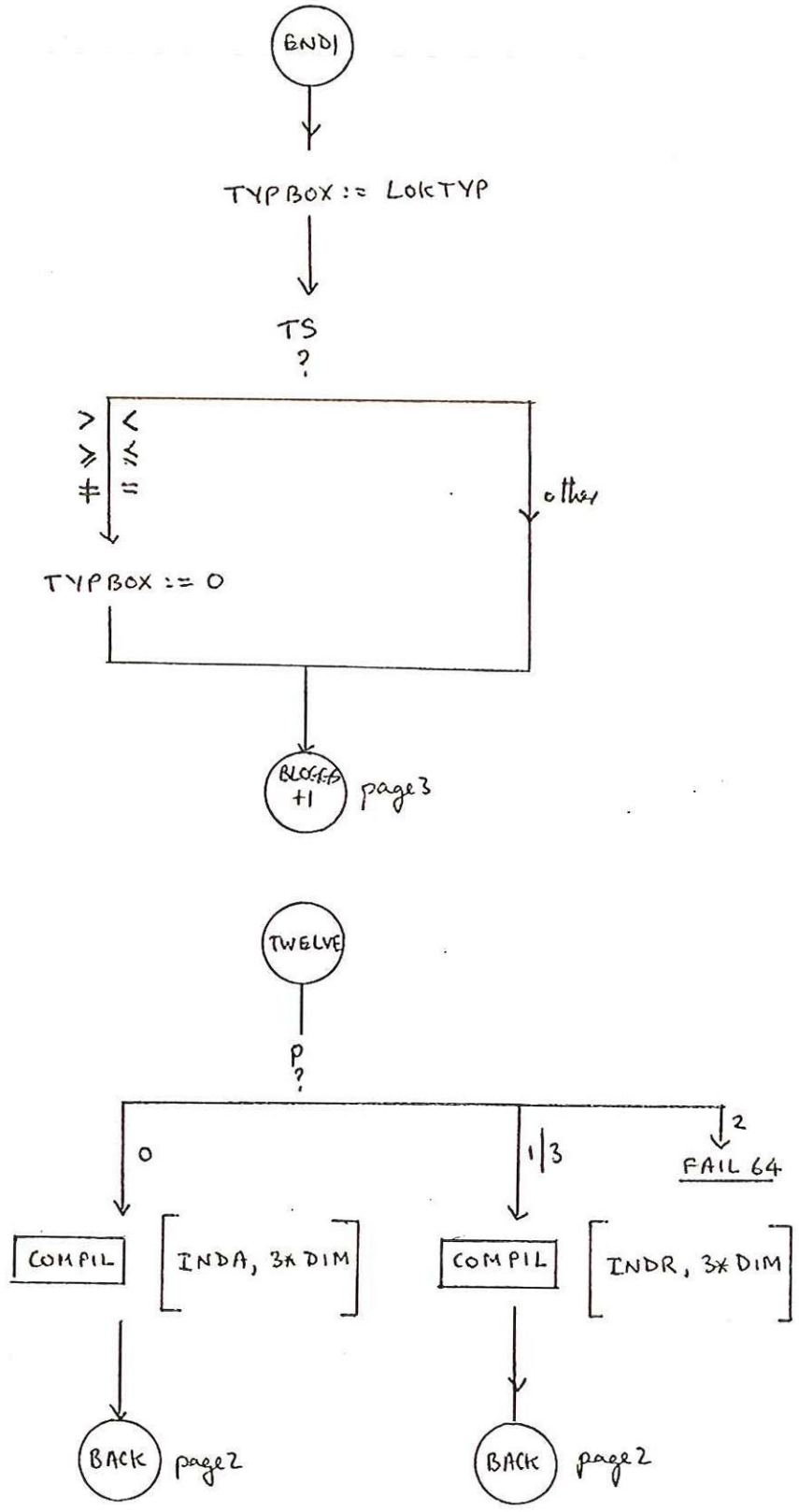
UNSTAK continued

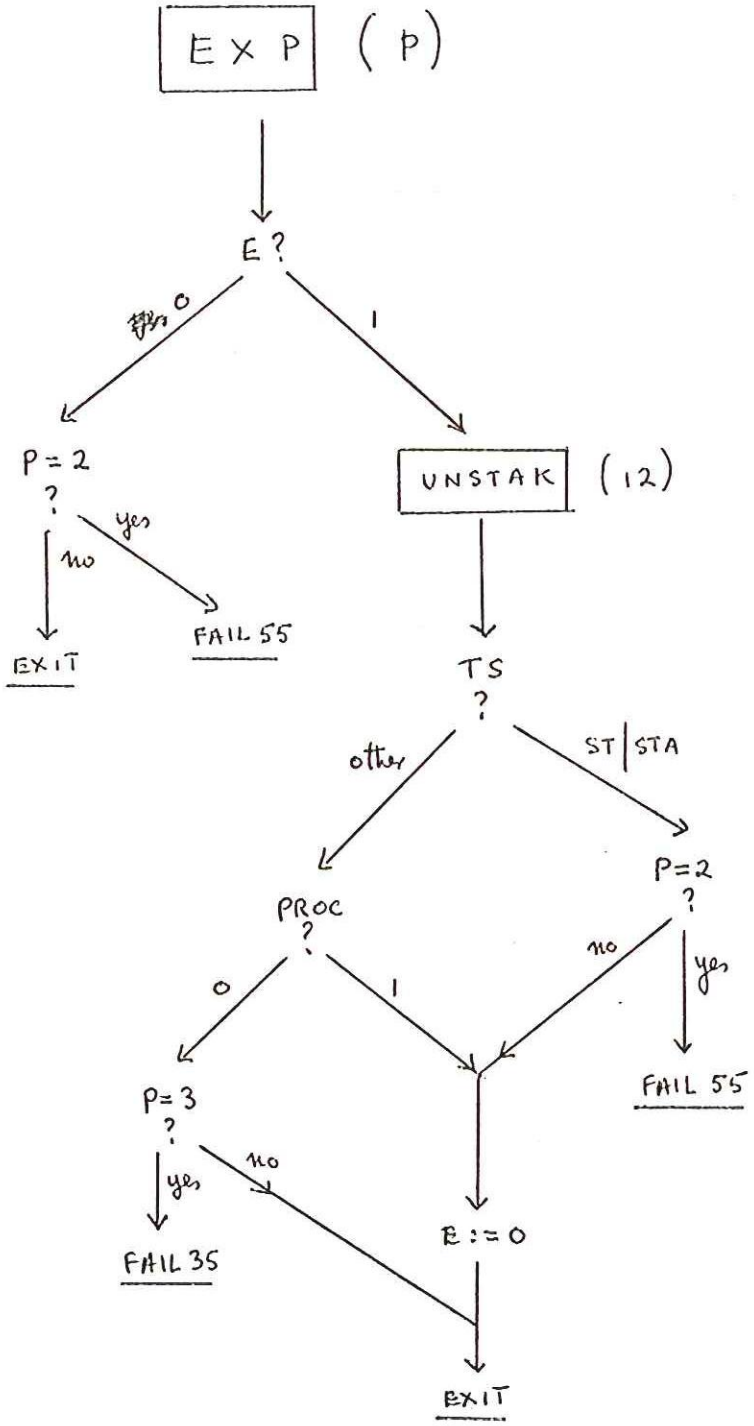


UNSTAK continued



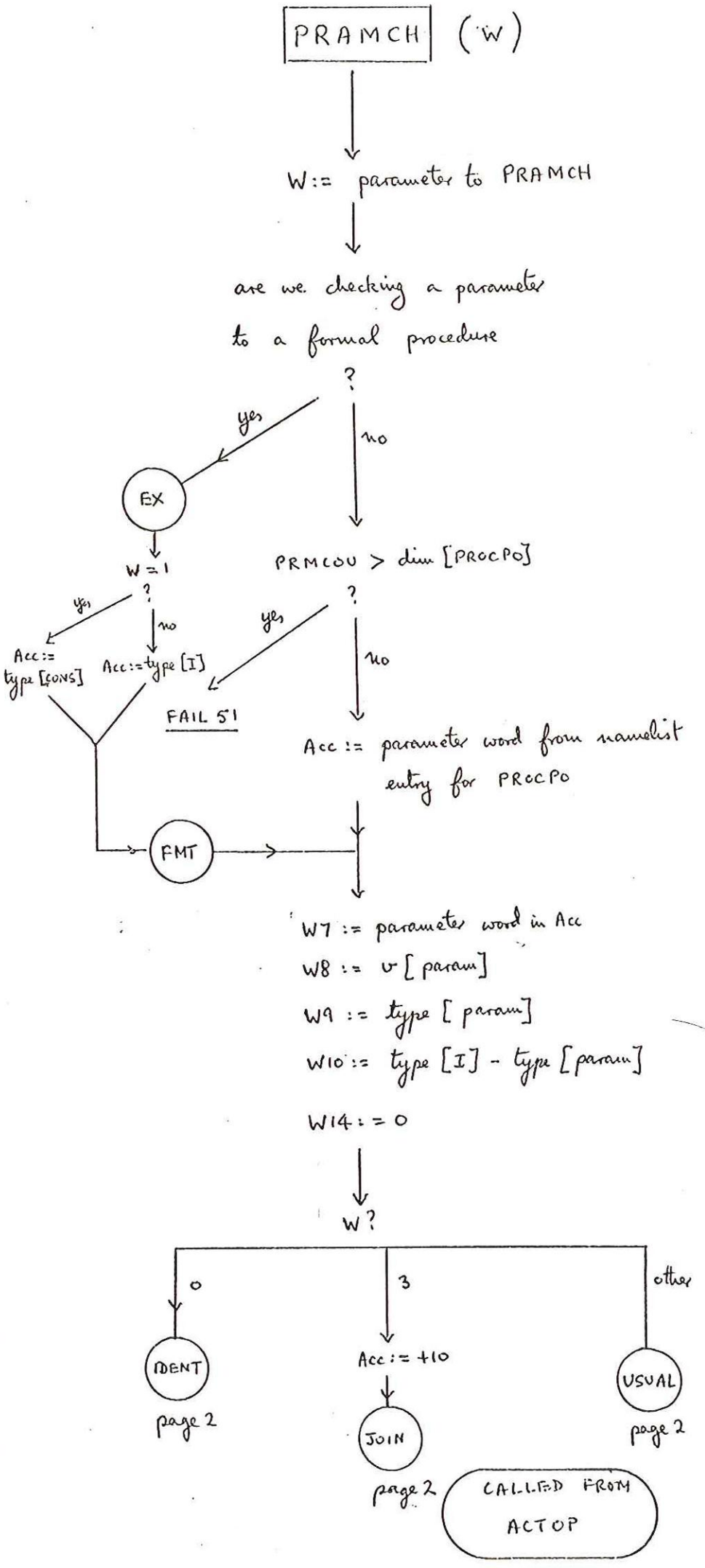
UNSTAK continued





CALLED FROM

FOR GOTO EF AOP RLT LOKOP COLON LRBRAK ENDSTA



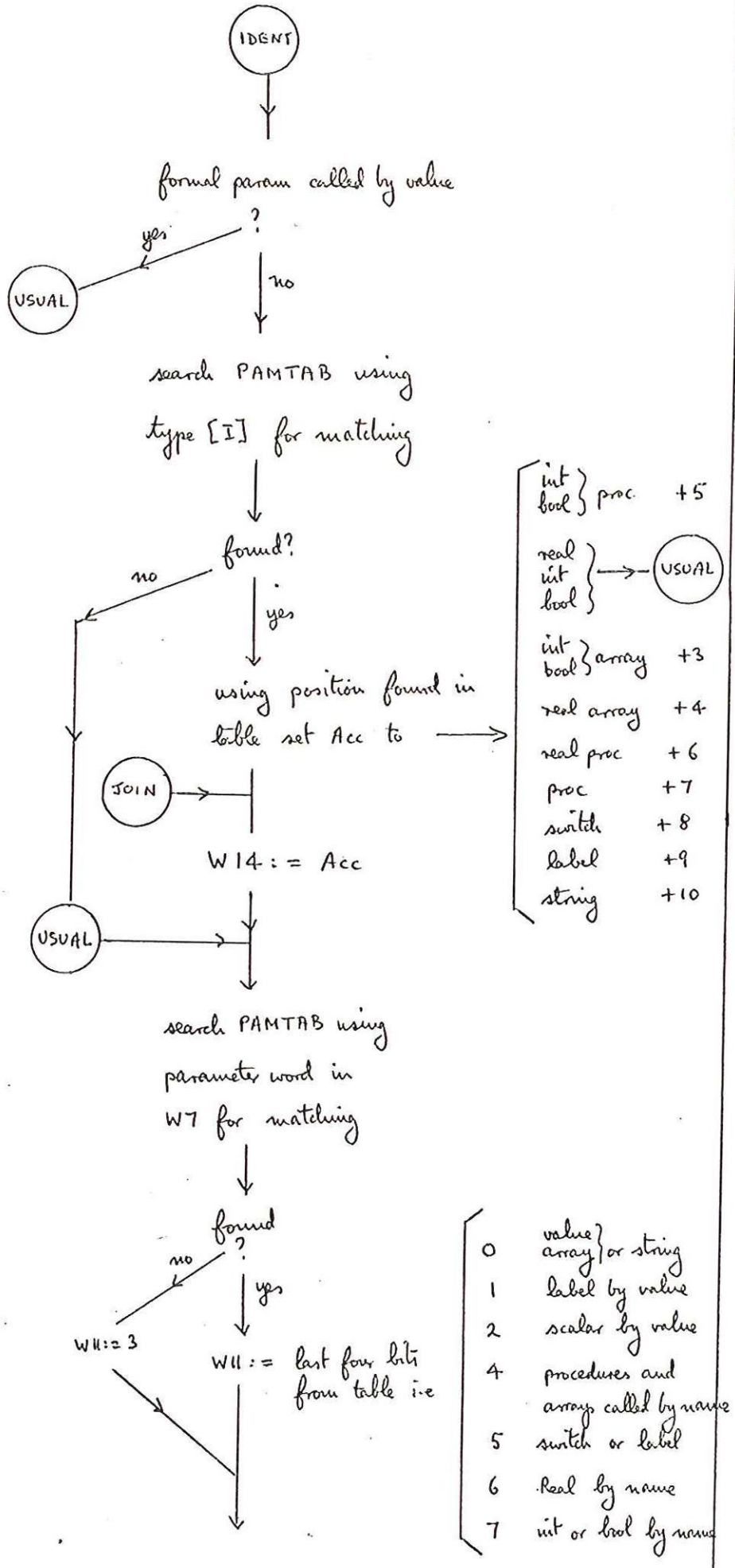
For formal procedures checking is done at run time : agreement is forced at this stage.

describes formal parameter

I points to actual param.

if W14 non zero at exit then a checking primitive is compiled.

PRAMCH continued

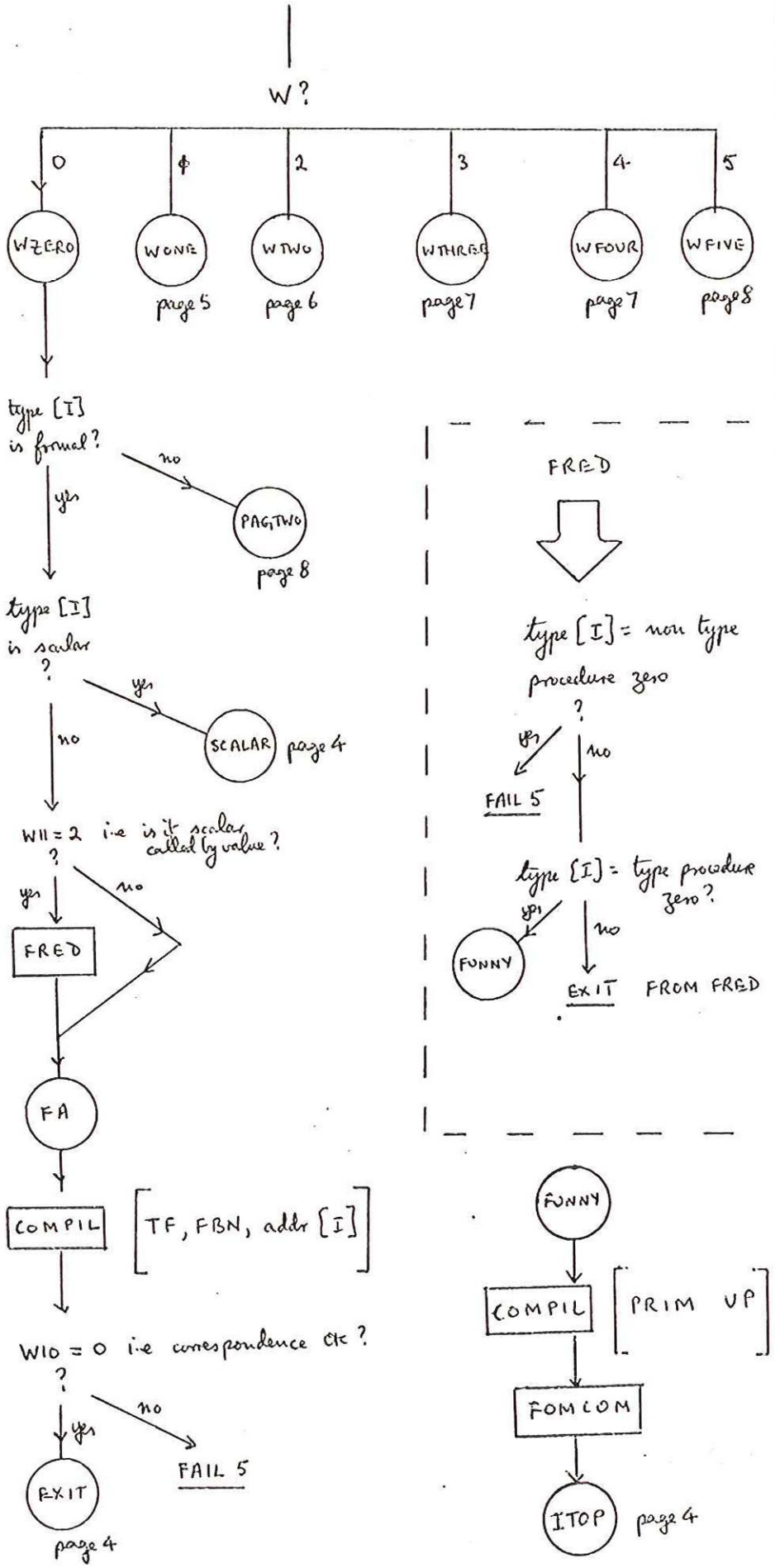


actual param search

W14 not altered in this case

formal param search

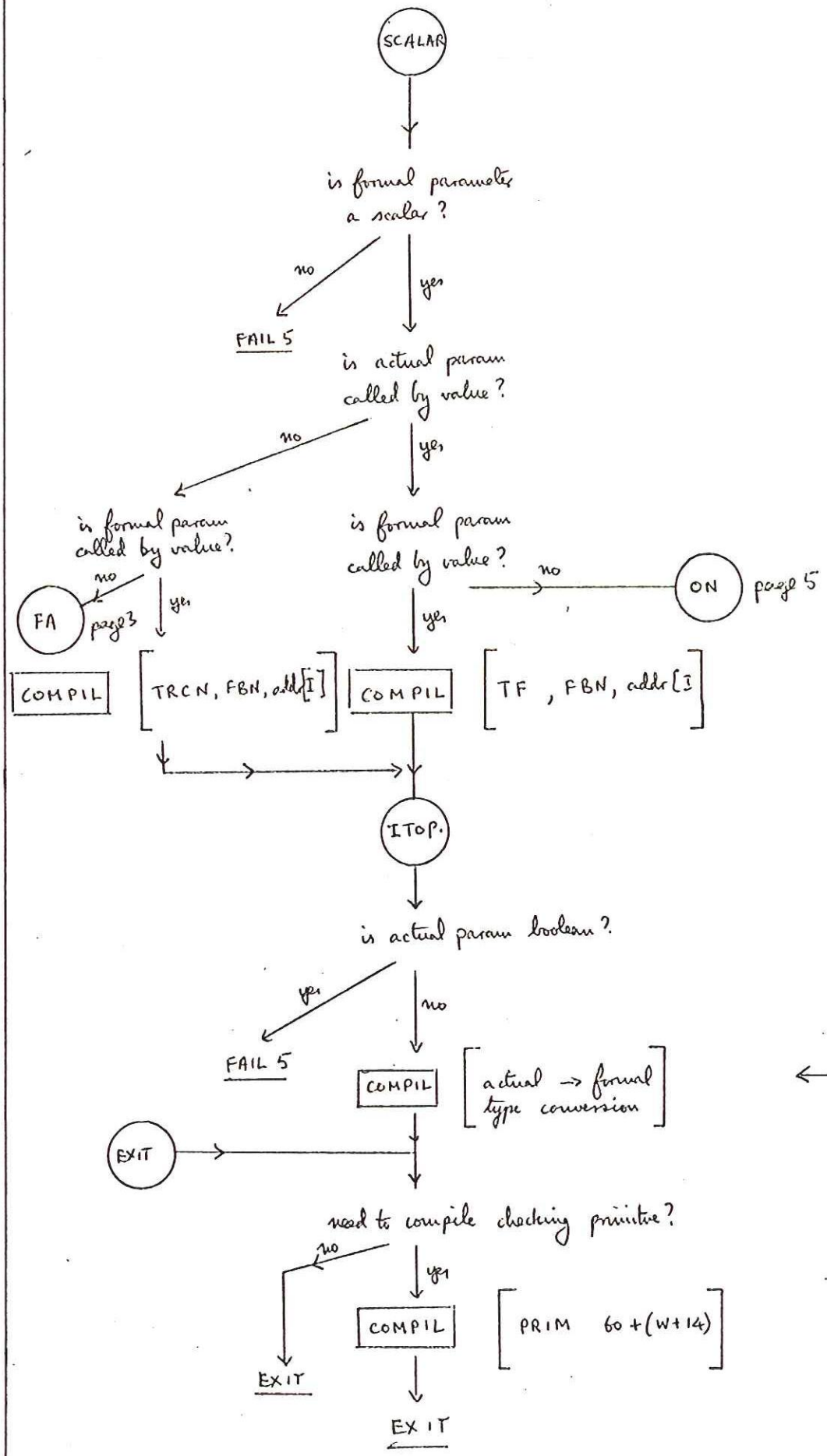
PRAMCH continued



local subroutine with exit to FUNNY

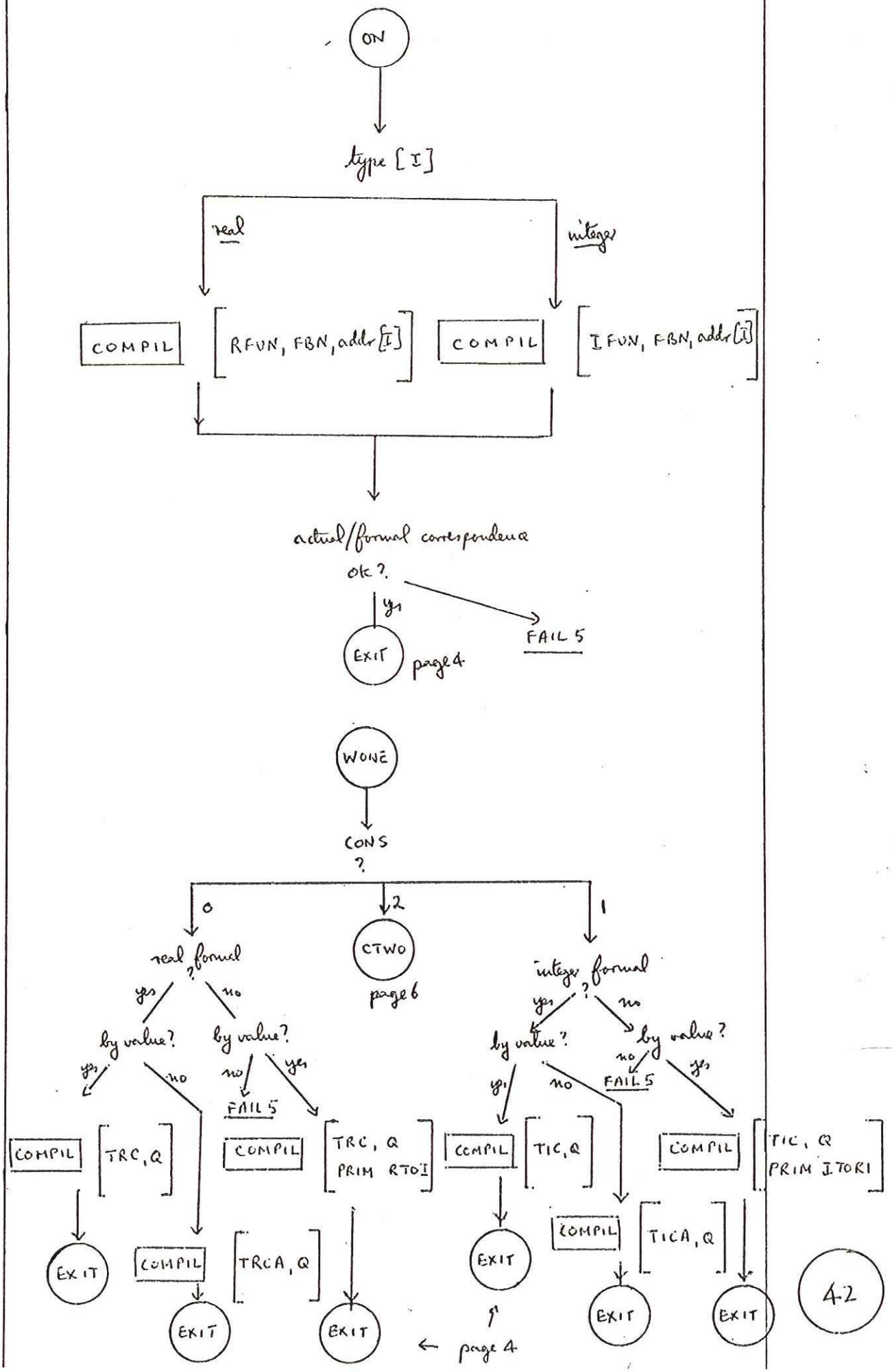
PRAMCH continued

a formal param is used here as an actual param

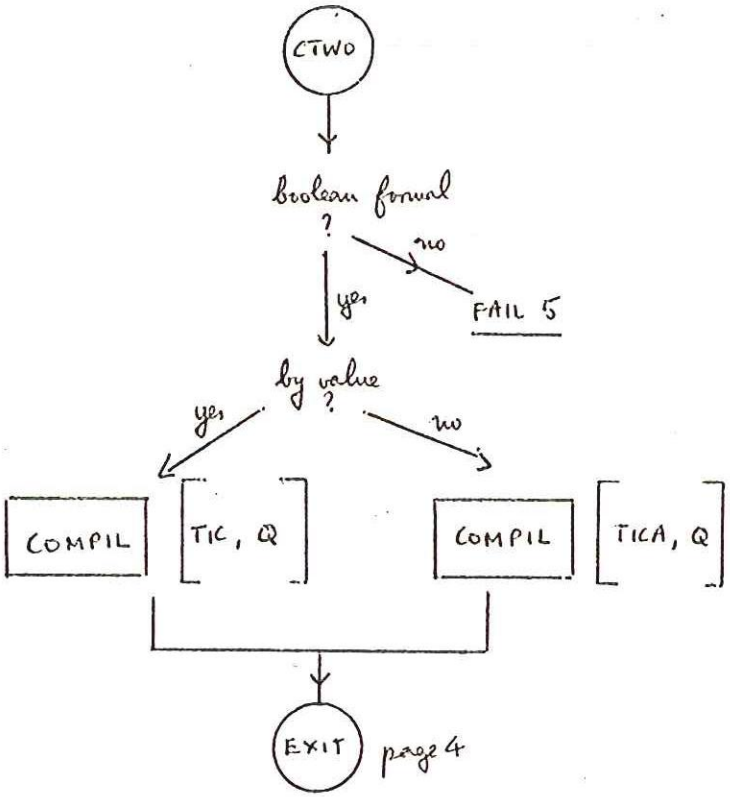


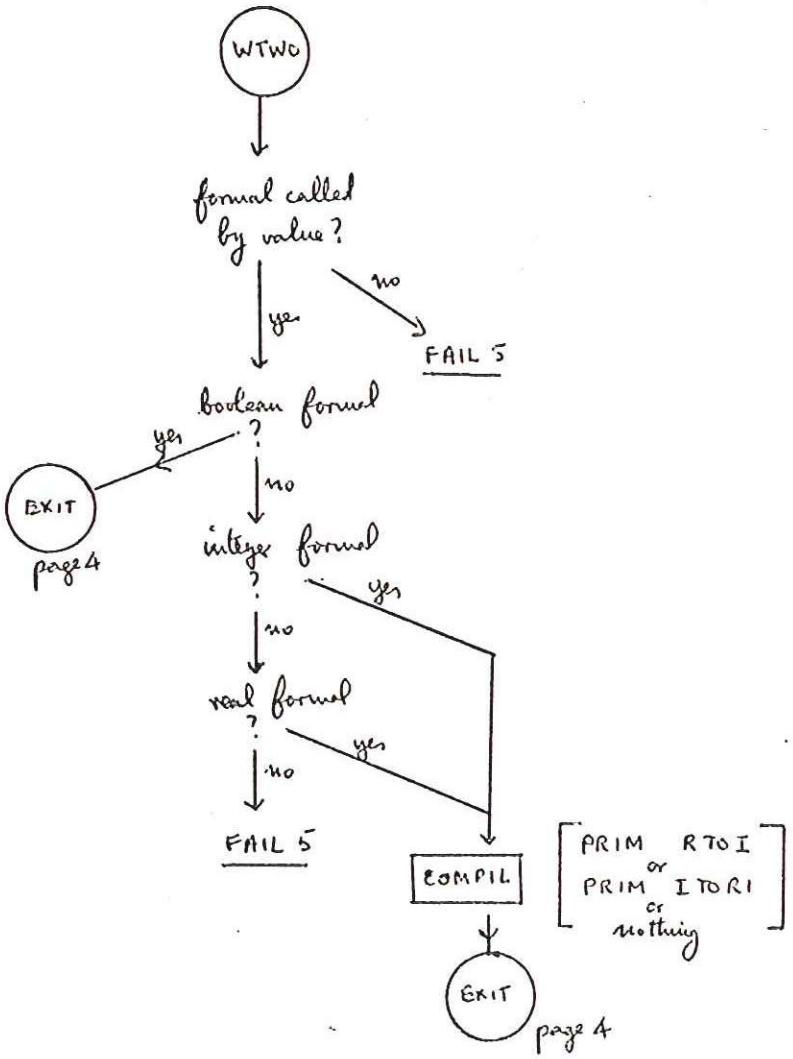
[if no conversion is necessary this is skipped]

PRAMCH continued



PRAMCH continued

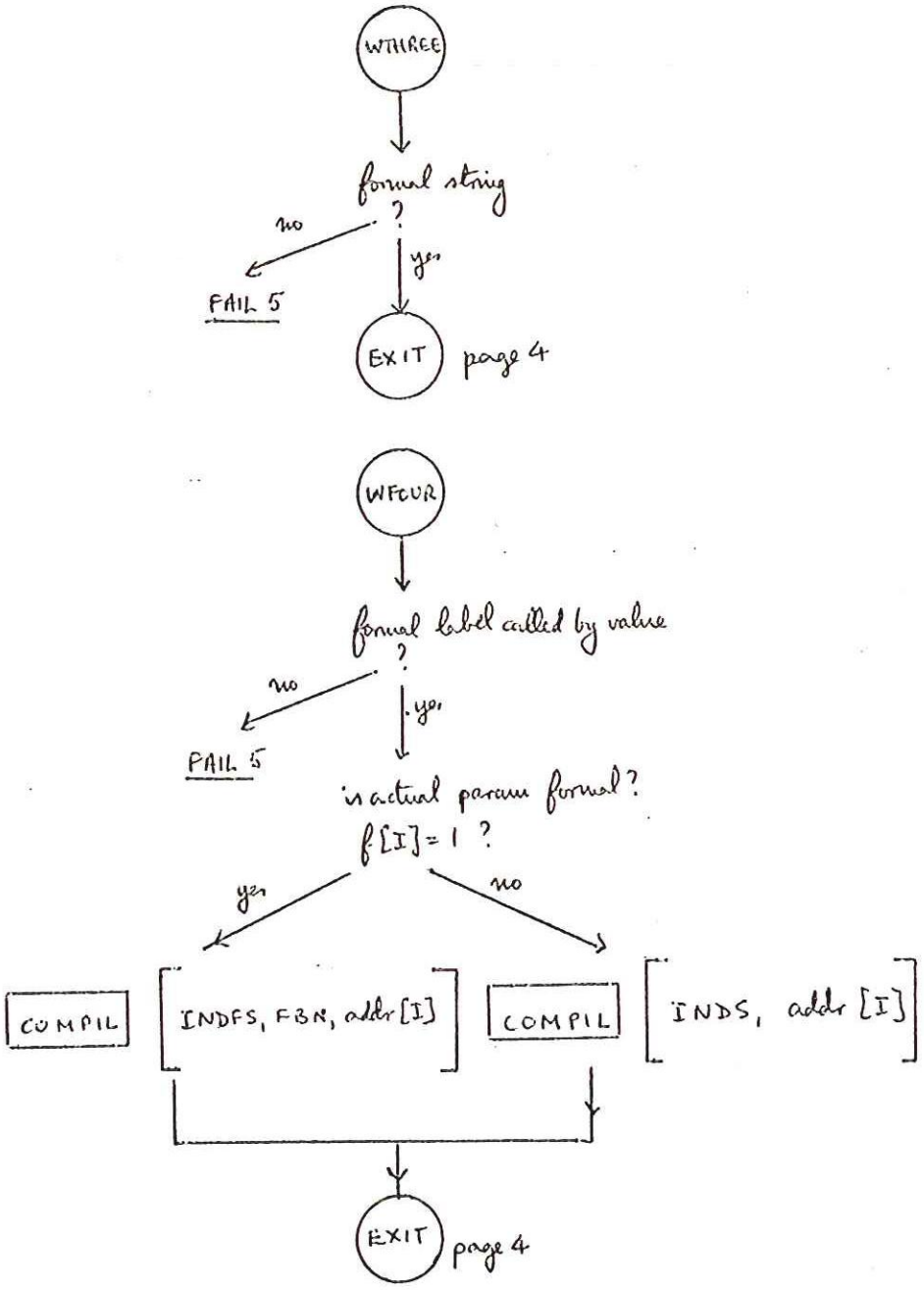




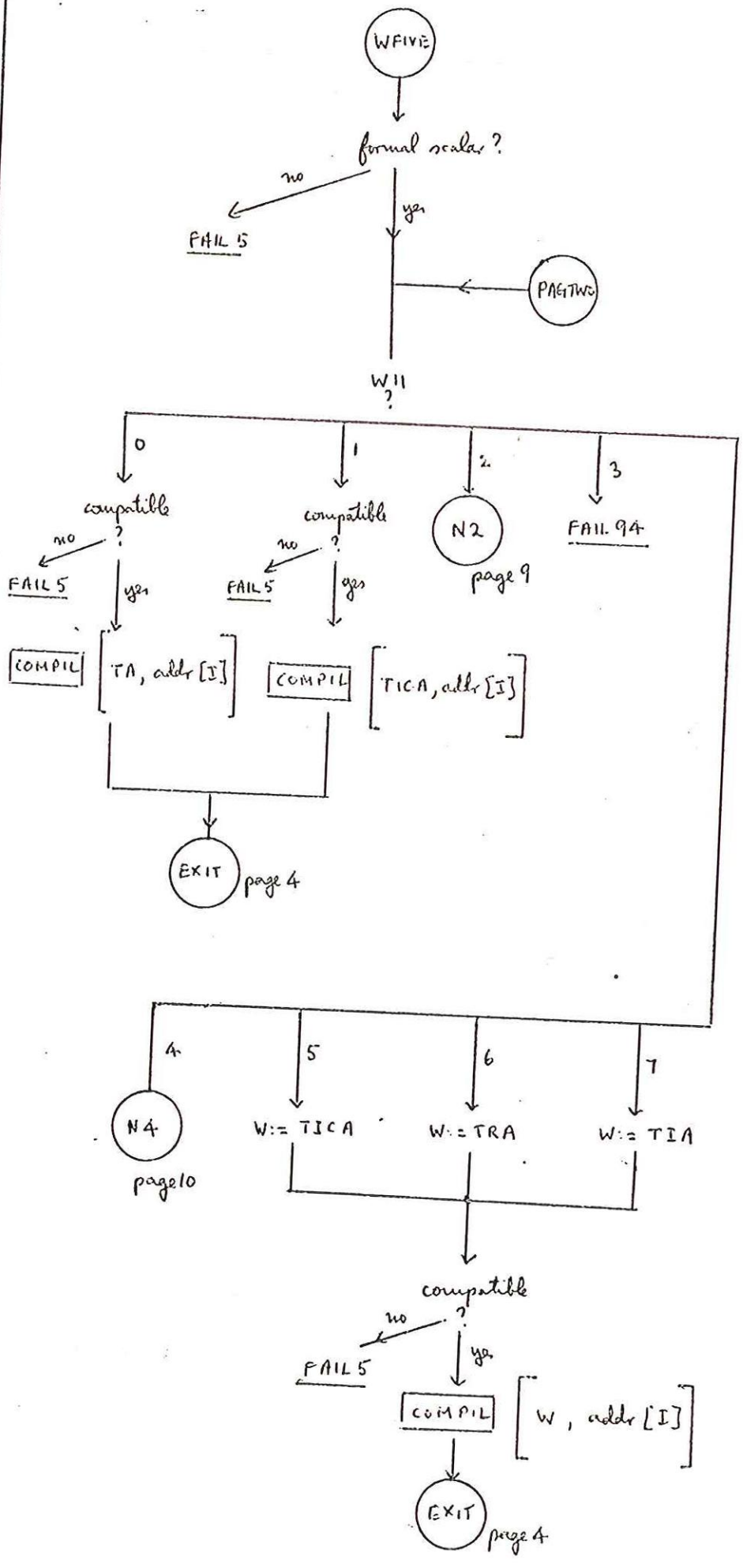
43

depends on TYPBOX to convert type from actual to formal

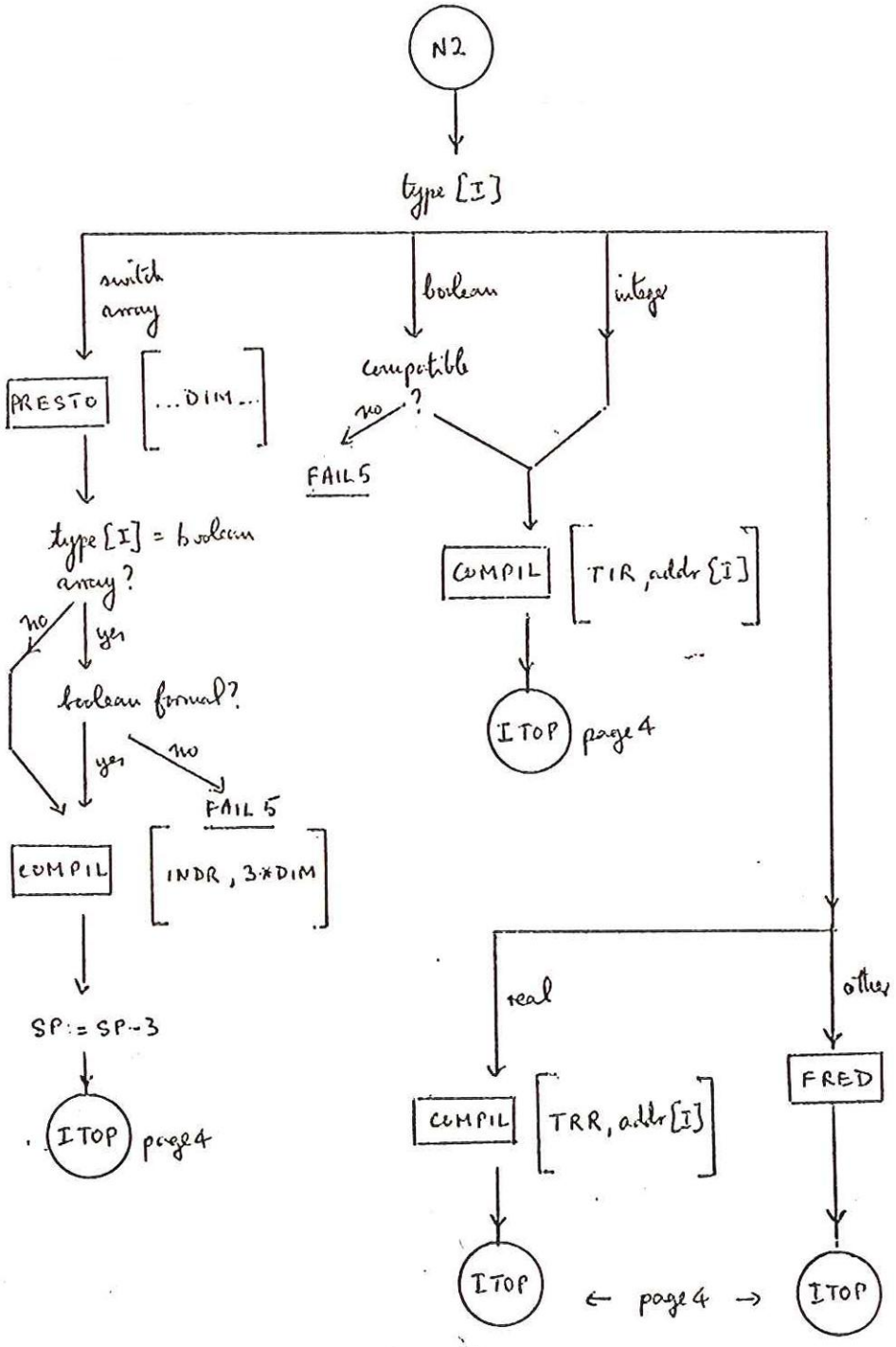
PRAMCH continued



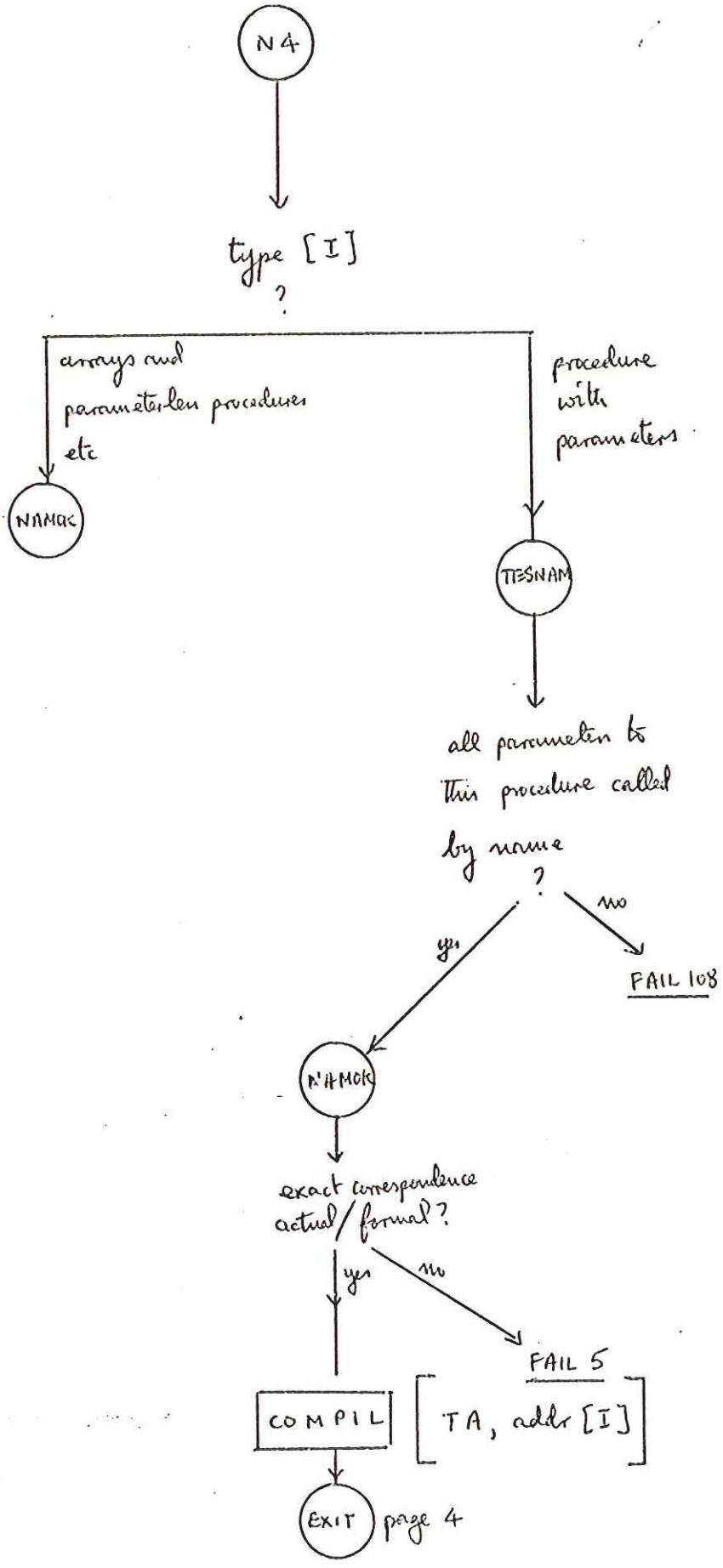
PRAMCH continued

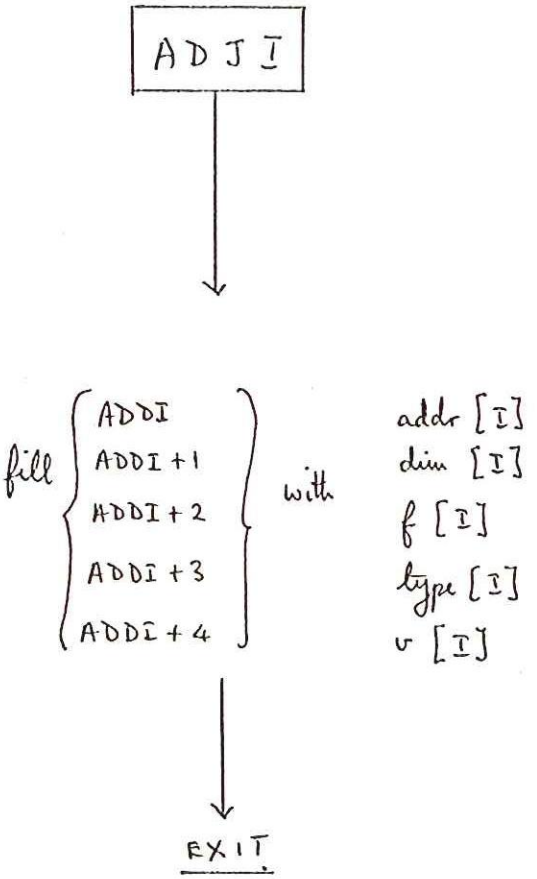


PRAMCH continued



PRAMCH continued



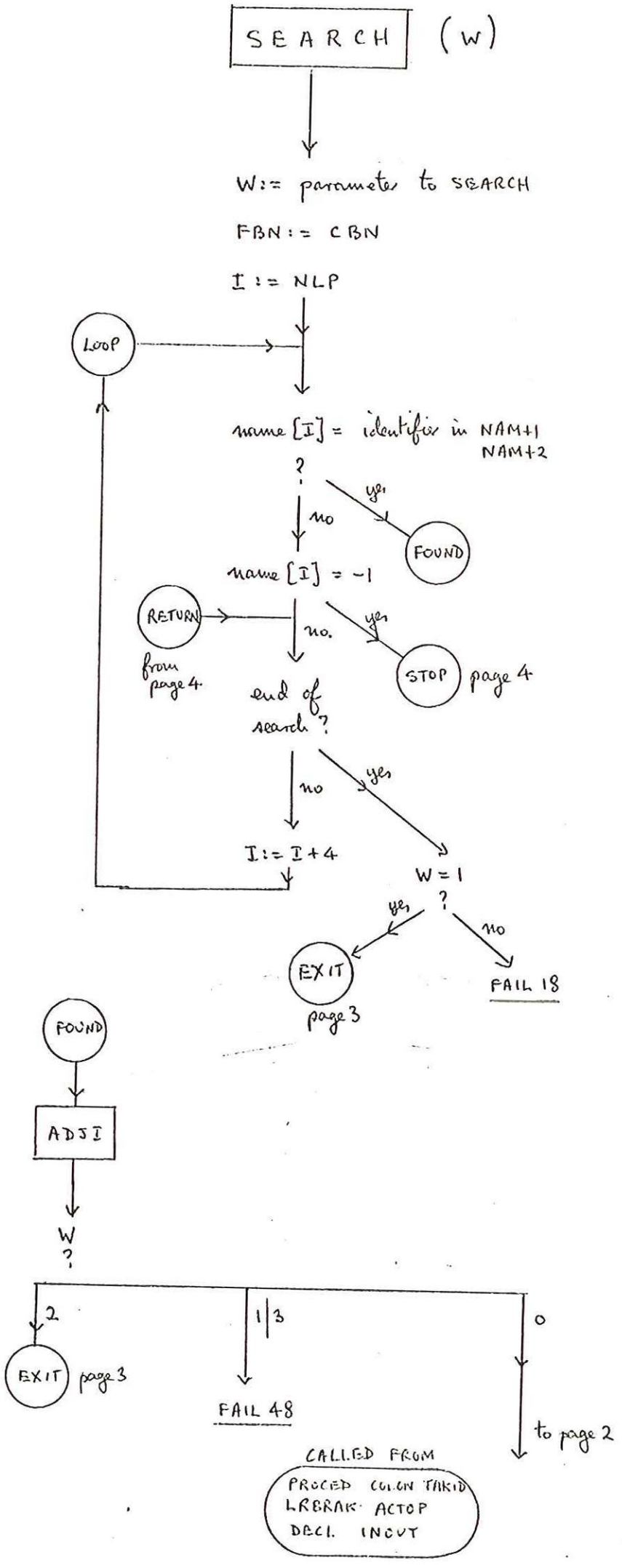


f = 1 if formal

v = 1 if called by value

CALLED FROM

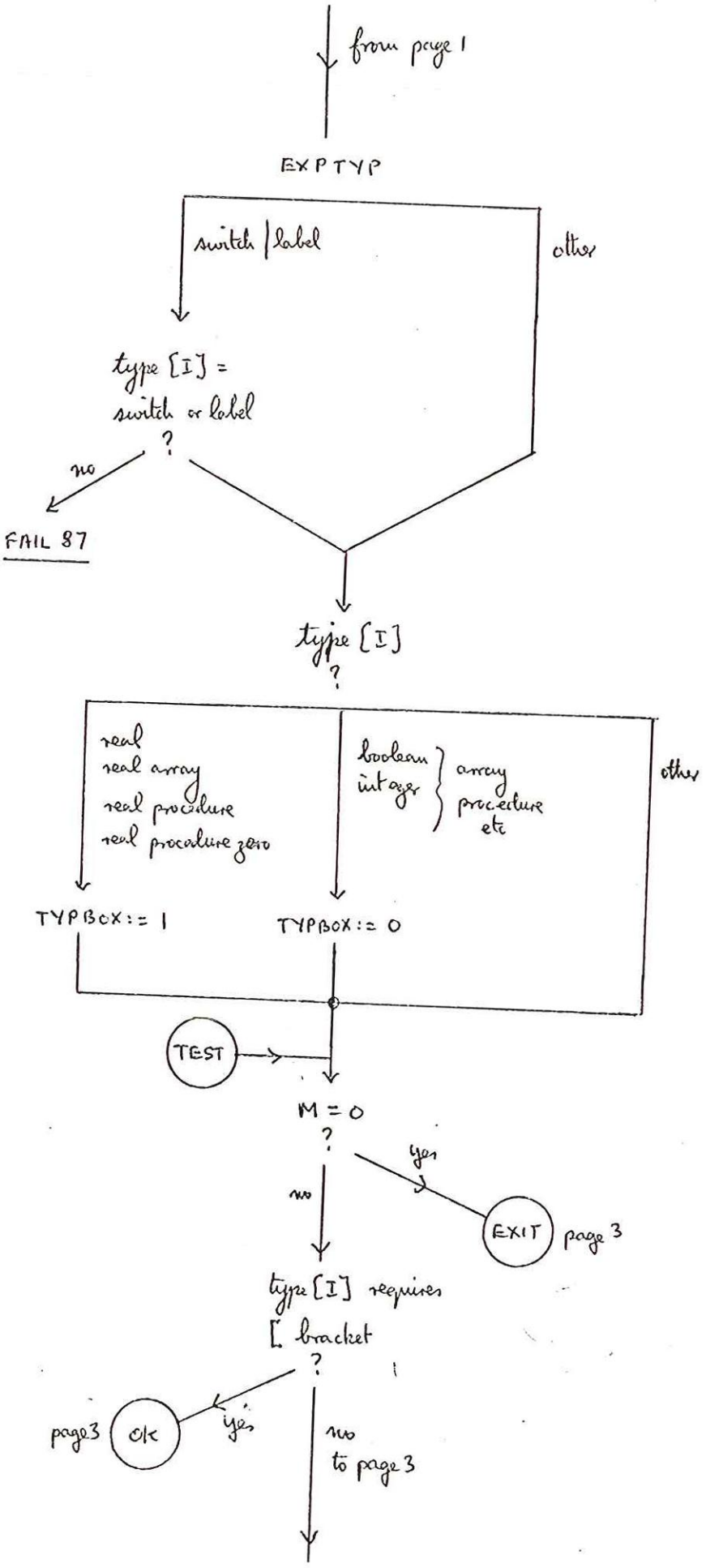
RRBRAK
SEARCH



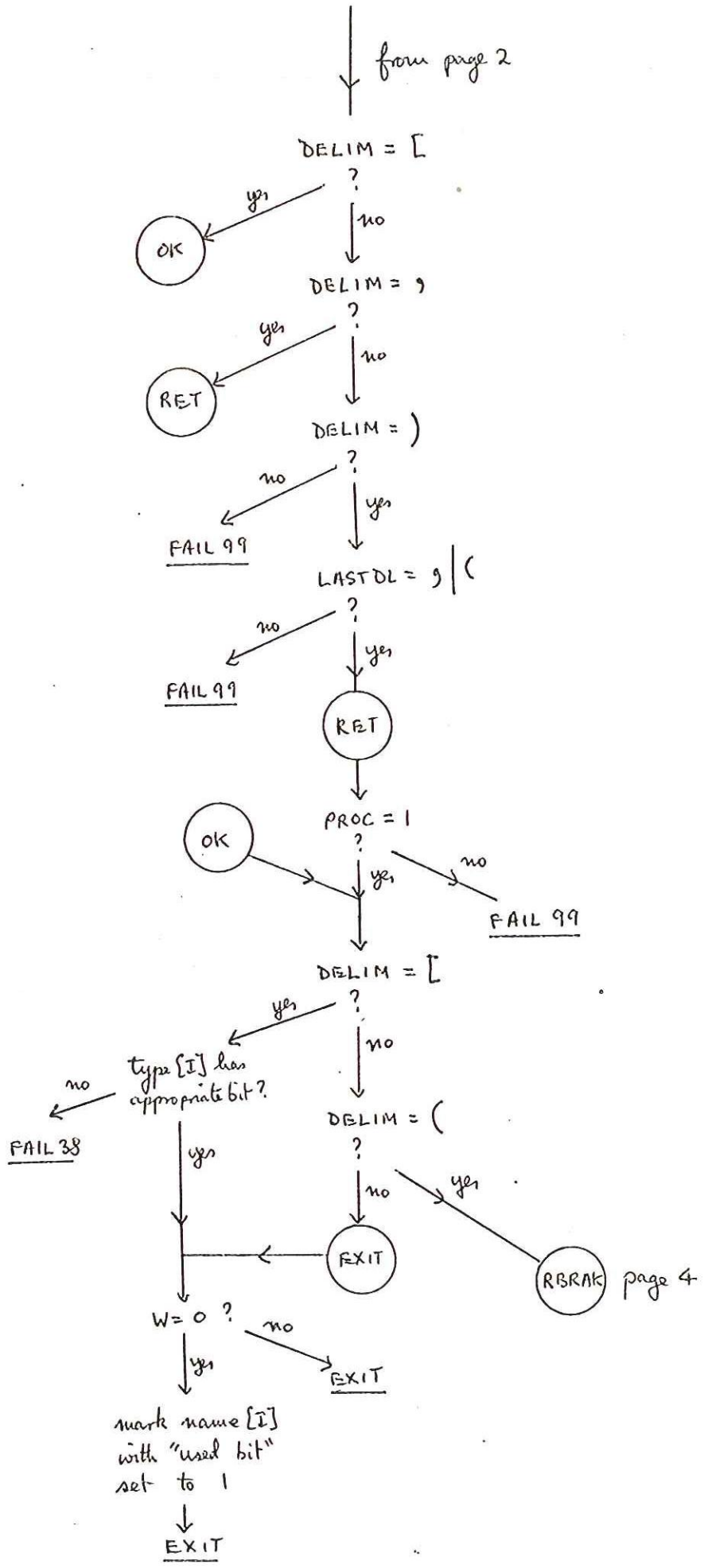
-1 represents stopper

search ends at 7995

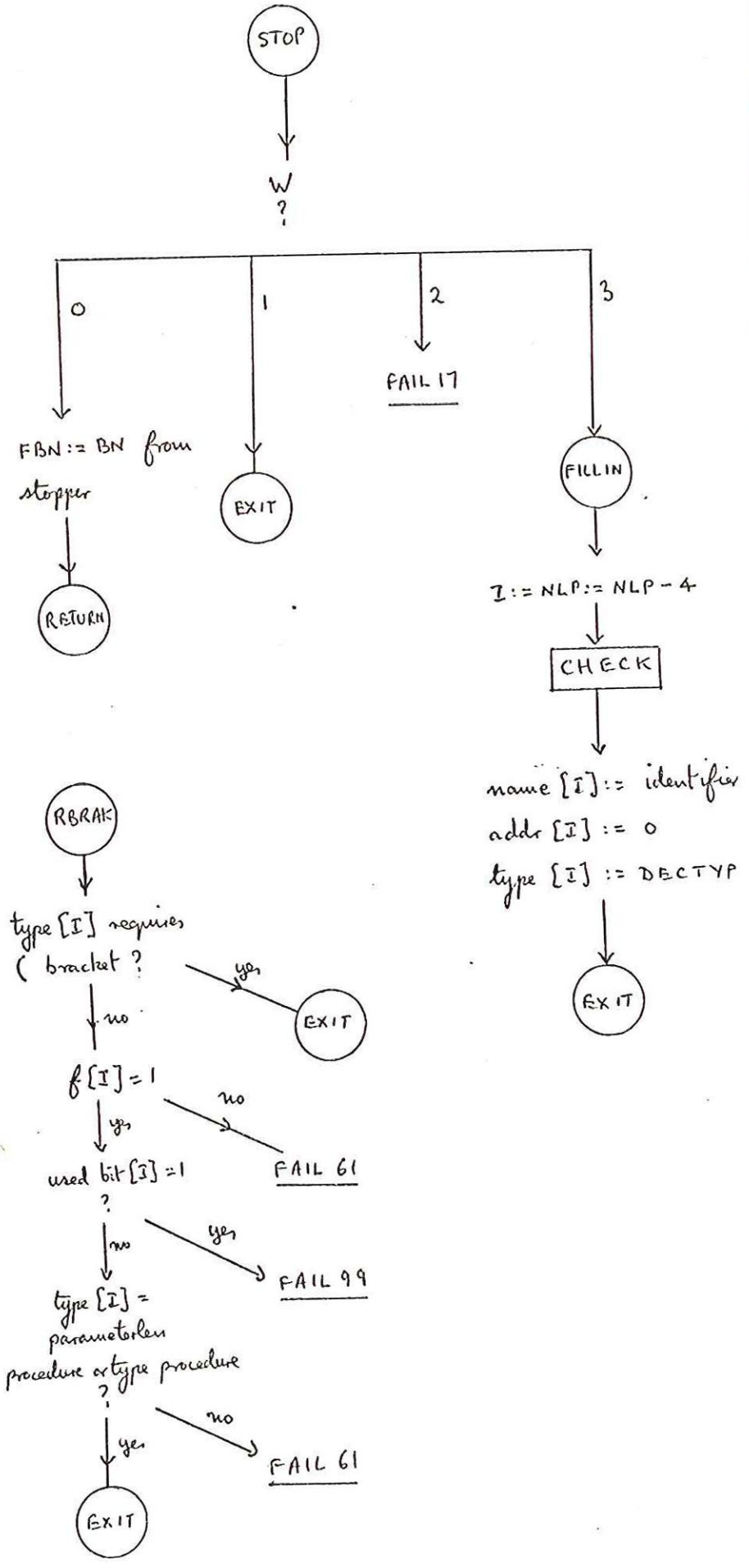
SEARCH continued

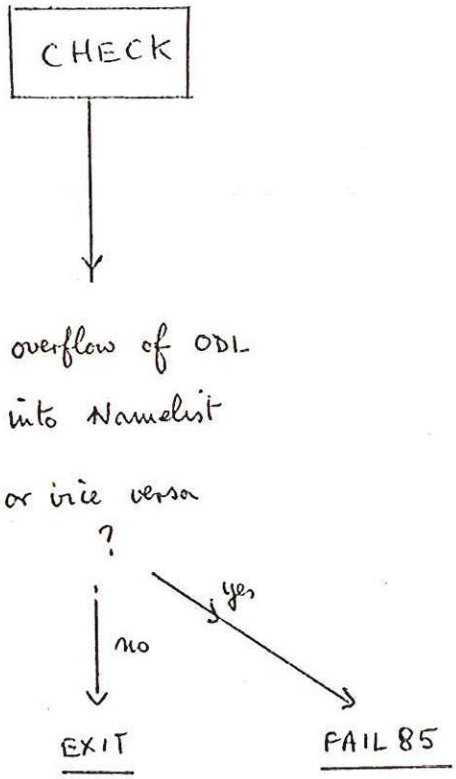


SEARCH continued

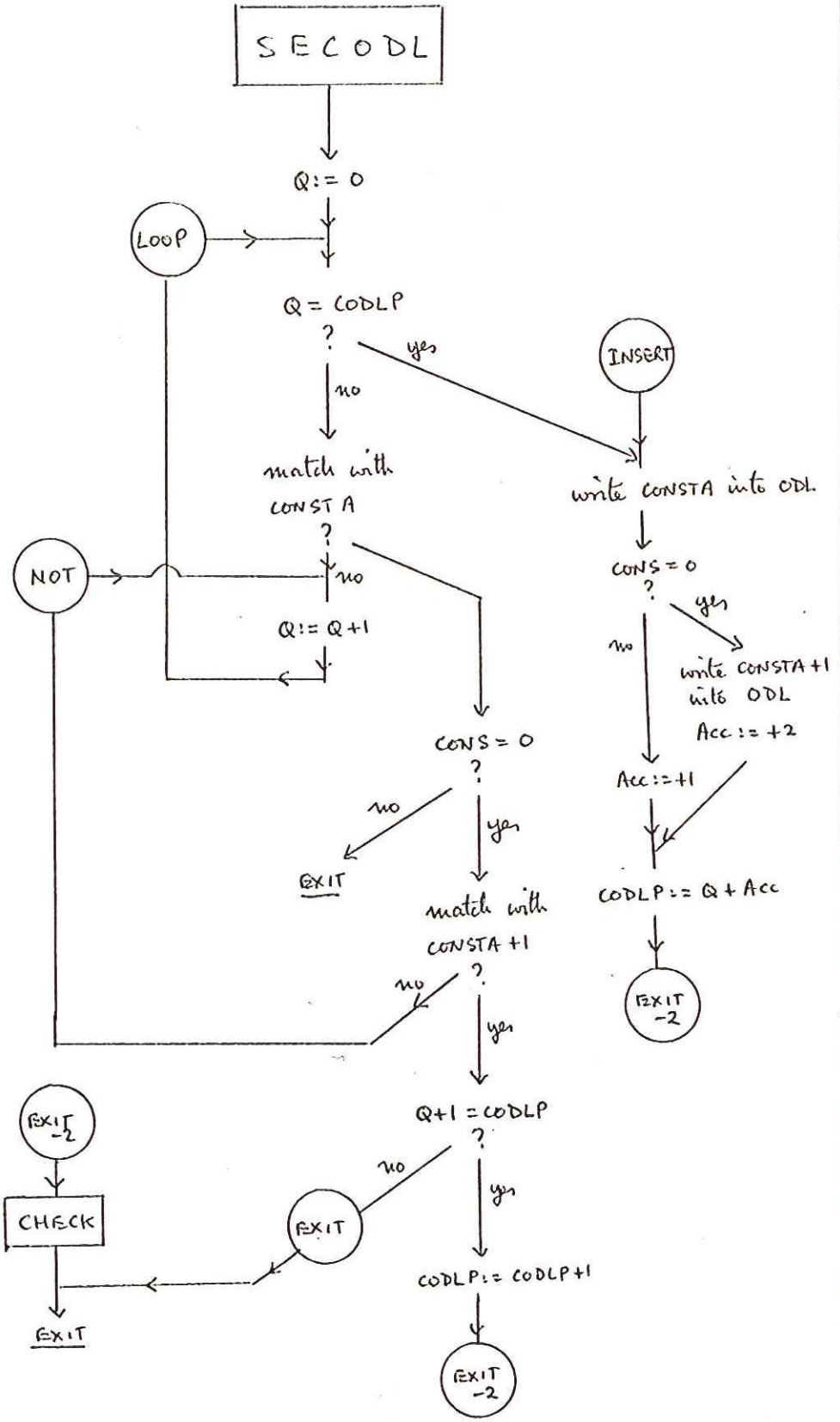


SEARCH continued





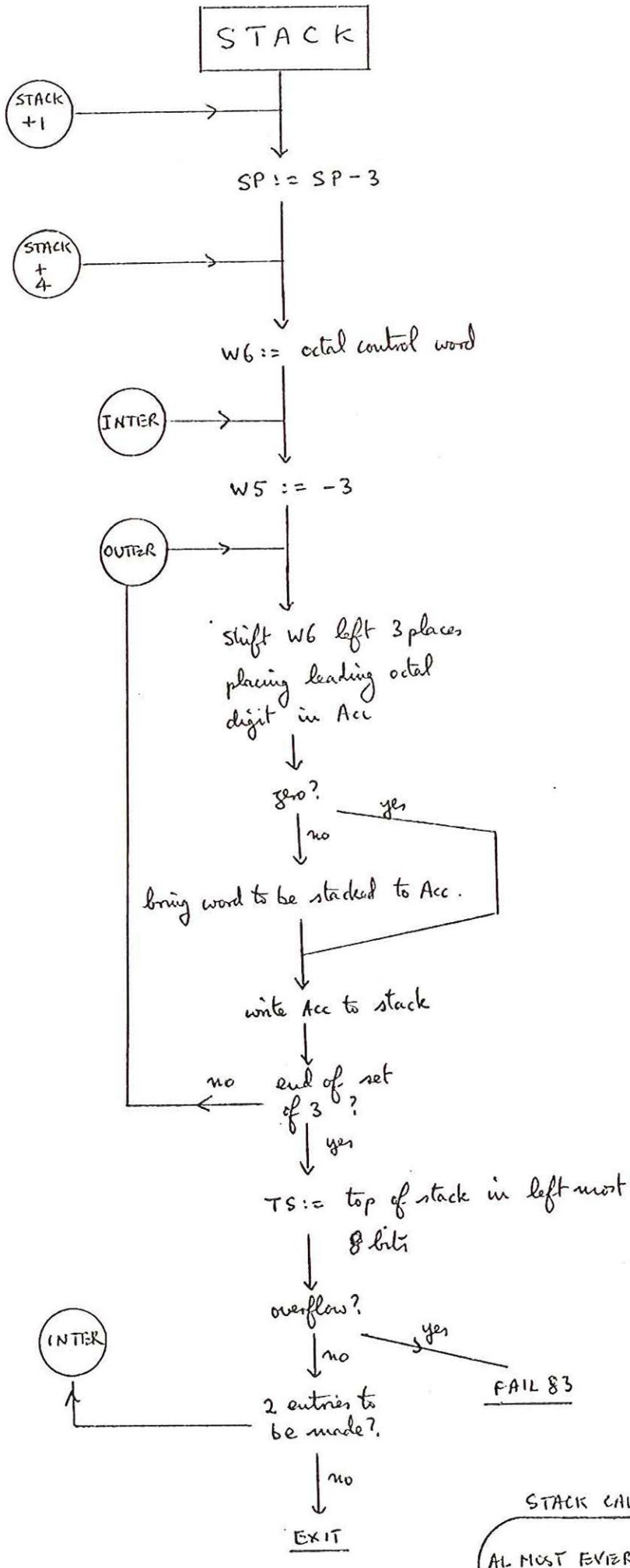
CALLED FROM
FOR SEARCH
SECODL DEC



real constant in CONSTA and CONSTA+1

CALLED FROM

TAKE ACTOP

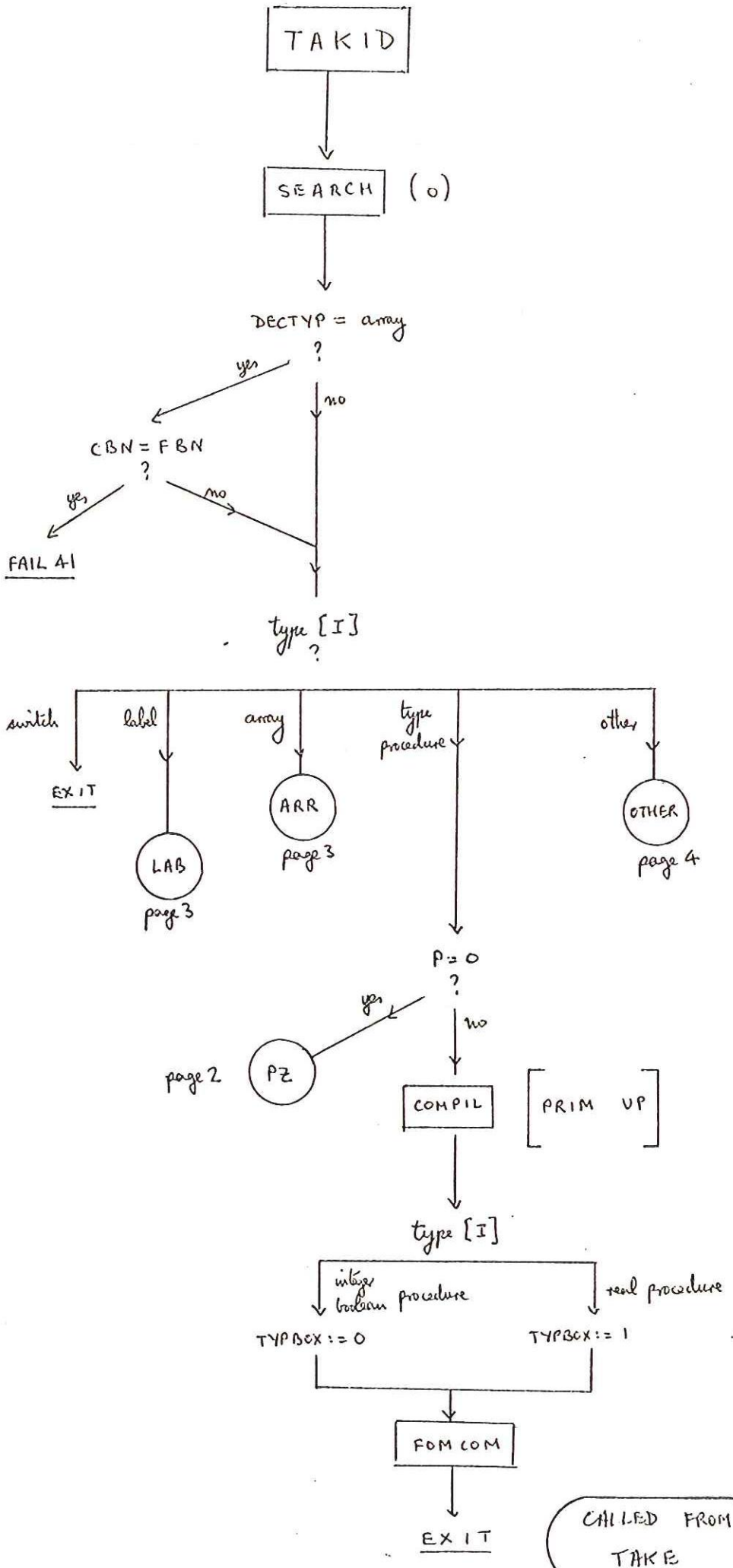


Overwrites top of stack with new entry

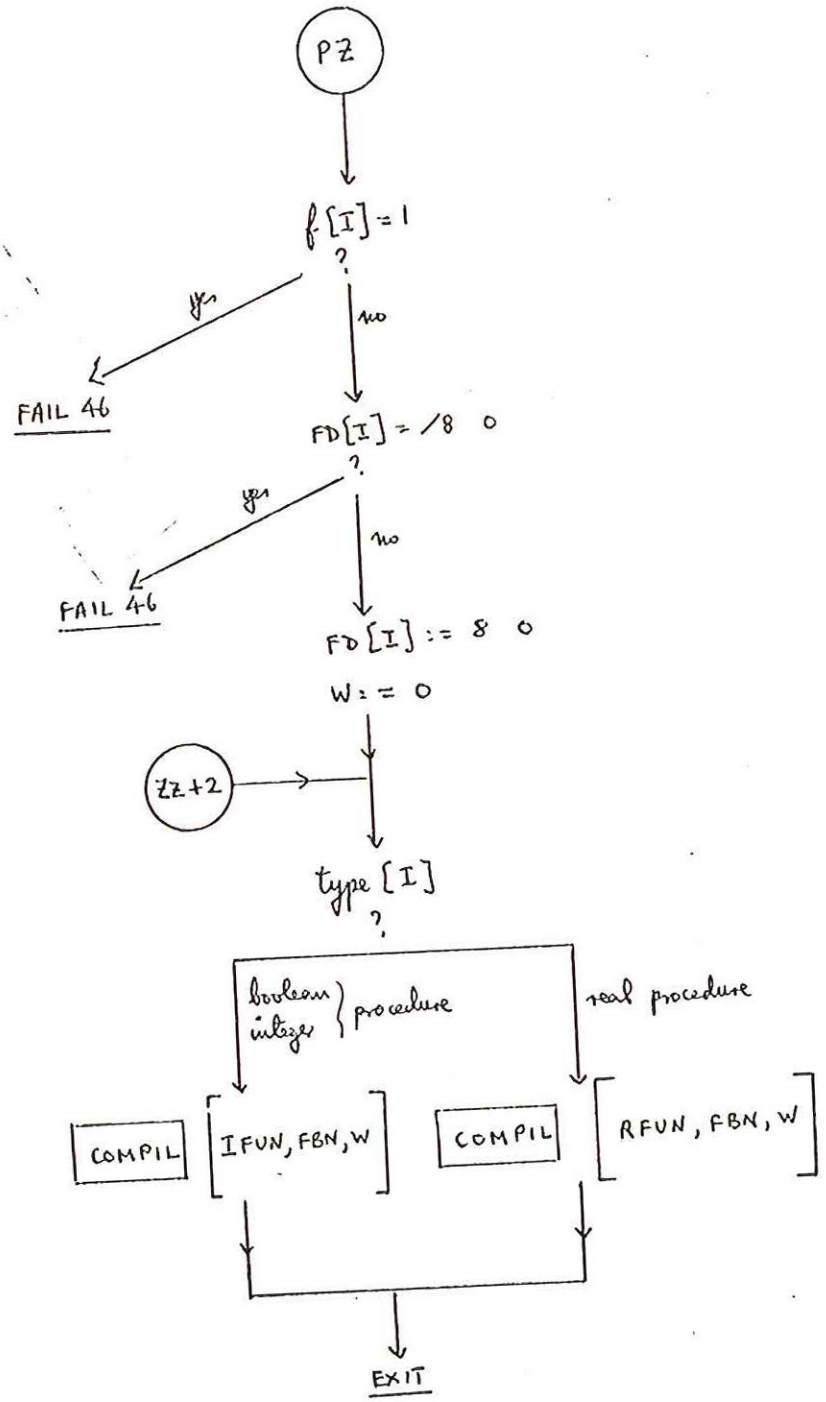
Normal Entry

STACK CALLED FROM

ALMOST EVERYWHERE

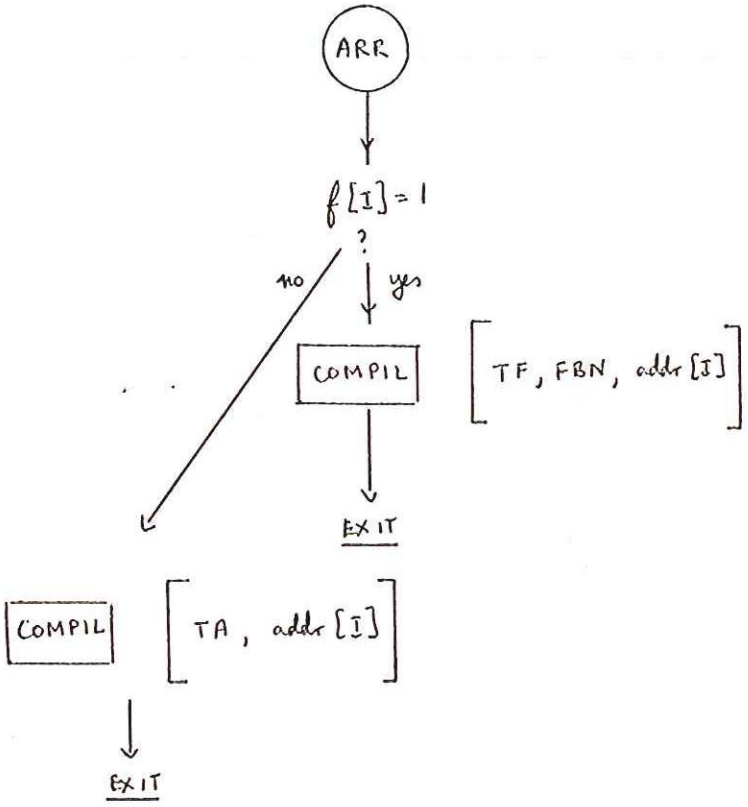

TAKID
SEARCH (0)
DECTYP = array ?
yes
no
CBN = FBN ?
yes
no
FAIL 41
type [I] ?
switch
label
array
type procedure
other
EXIT
LAB
page 3
ARR
page 3
OTHER
page 4
P = 0 ?
yes
no
page 2
PZ
COMPIL
[PRIM UP]
type [I]
integer boolean procedure
real procedure
TYPBOX := 0
TYPBOX := 1
FOM COM
EXIT
CALLED FROM TAKE

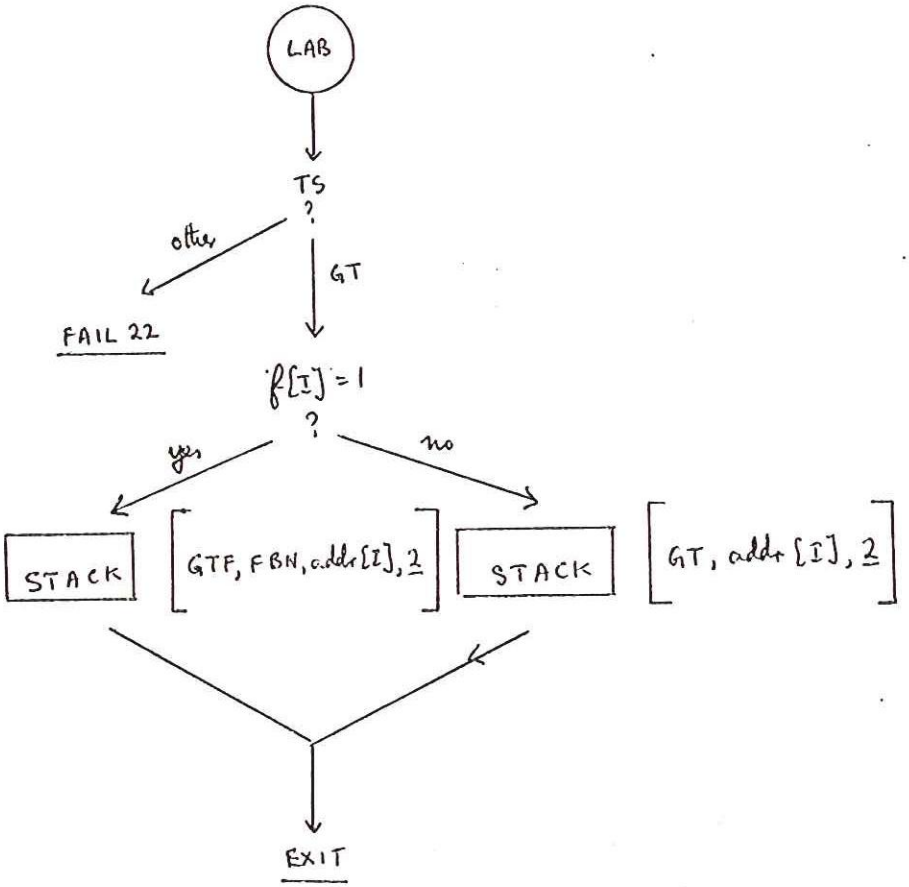
TAKID continued



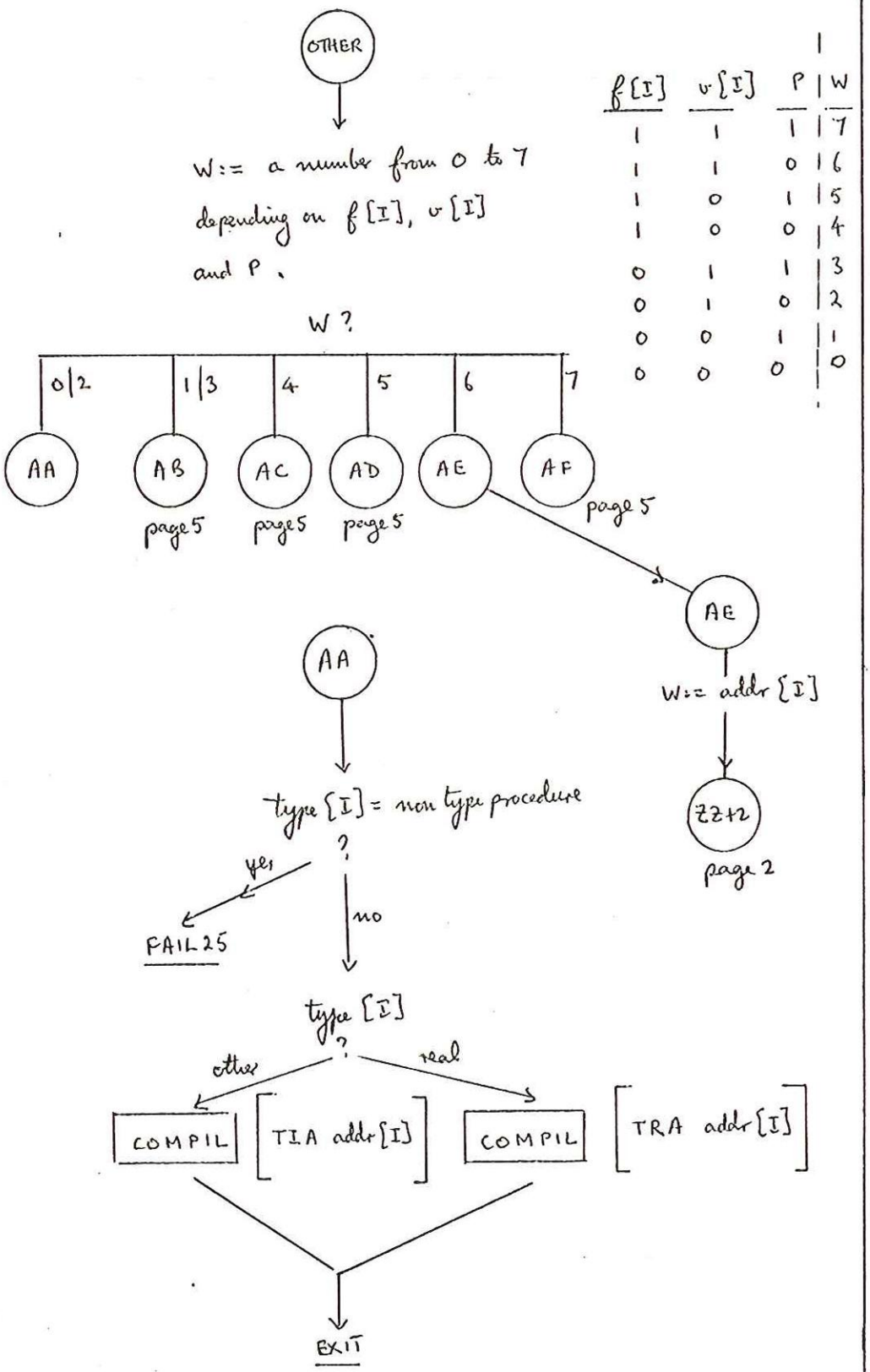
Fail if assignment to function name outside procedure body

TAKID continued



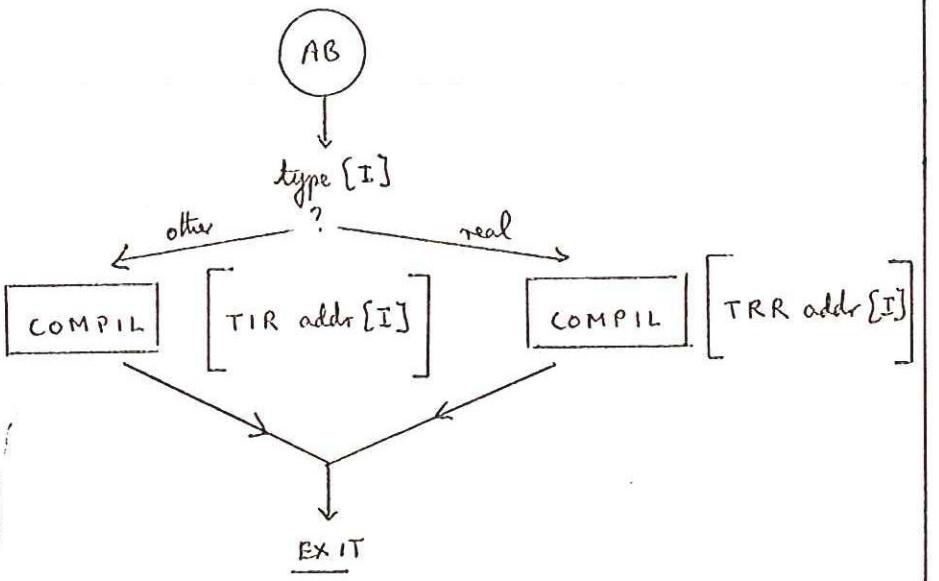


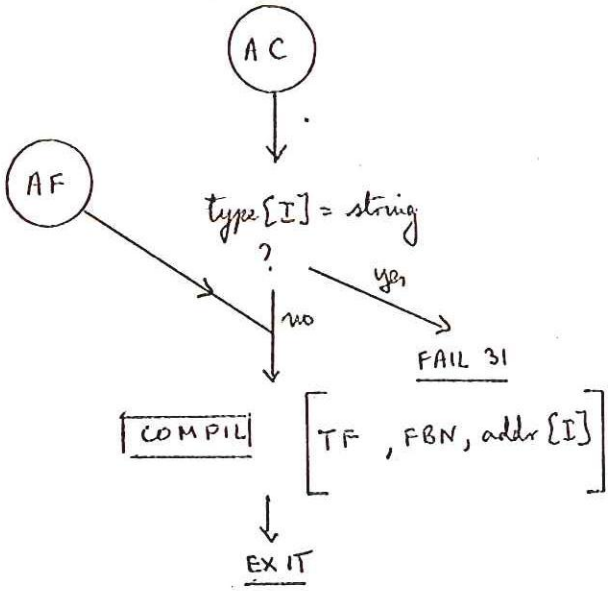
TAKID continued

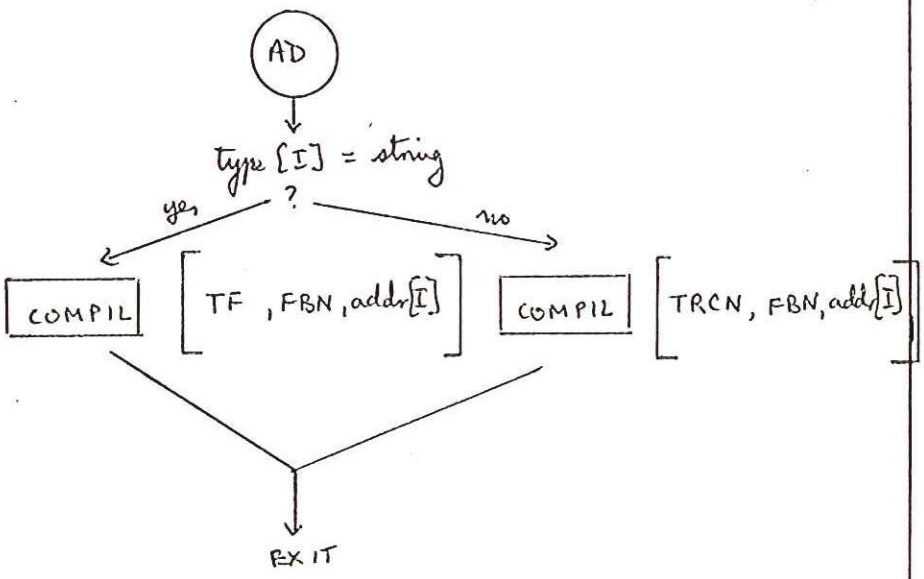


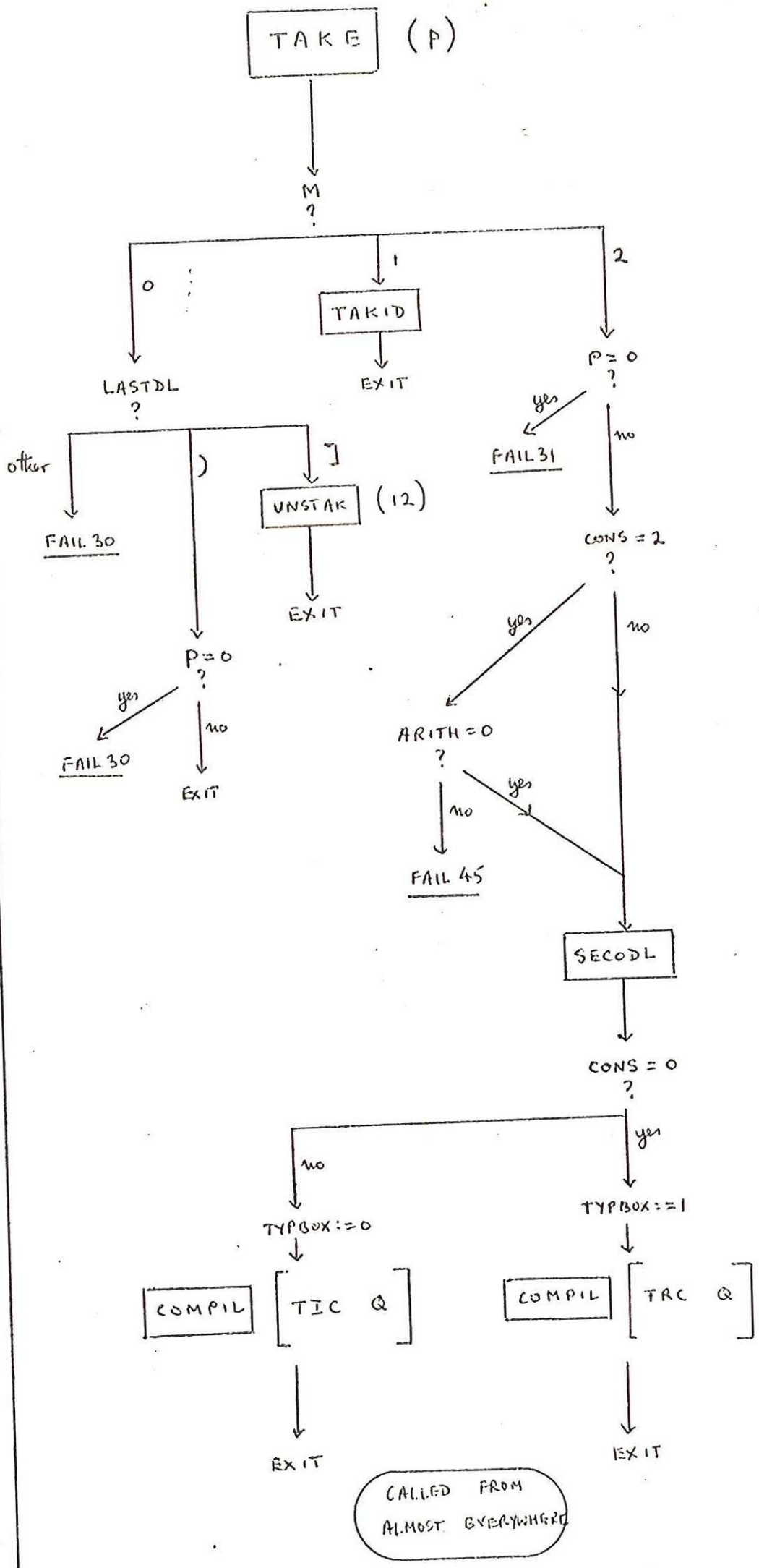
f[I]	v[I]	P	W
1	1	1	7
1	1	0	6
1	0	1	5
1	0	0	4
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0

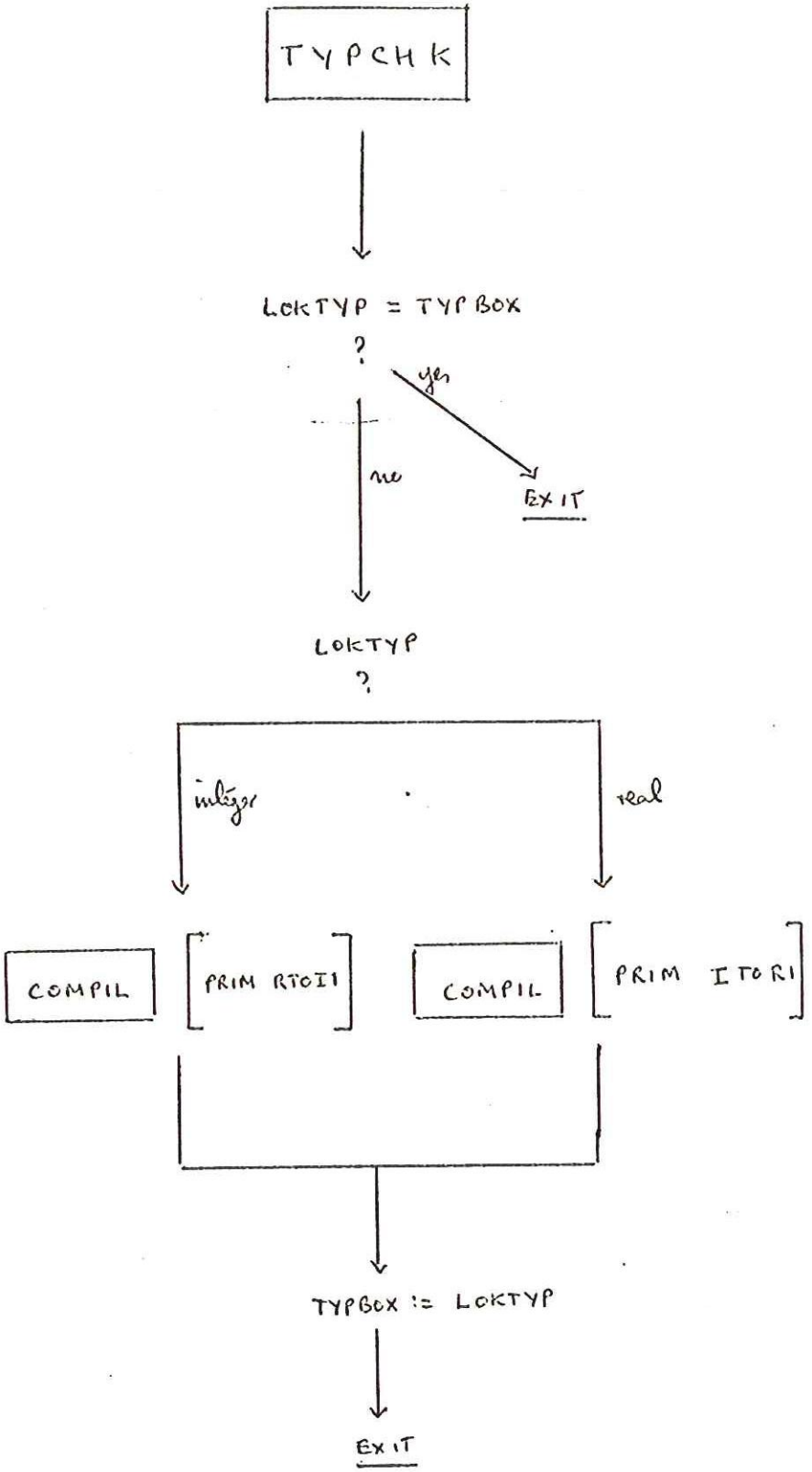
TAKID continued



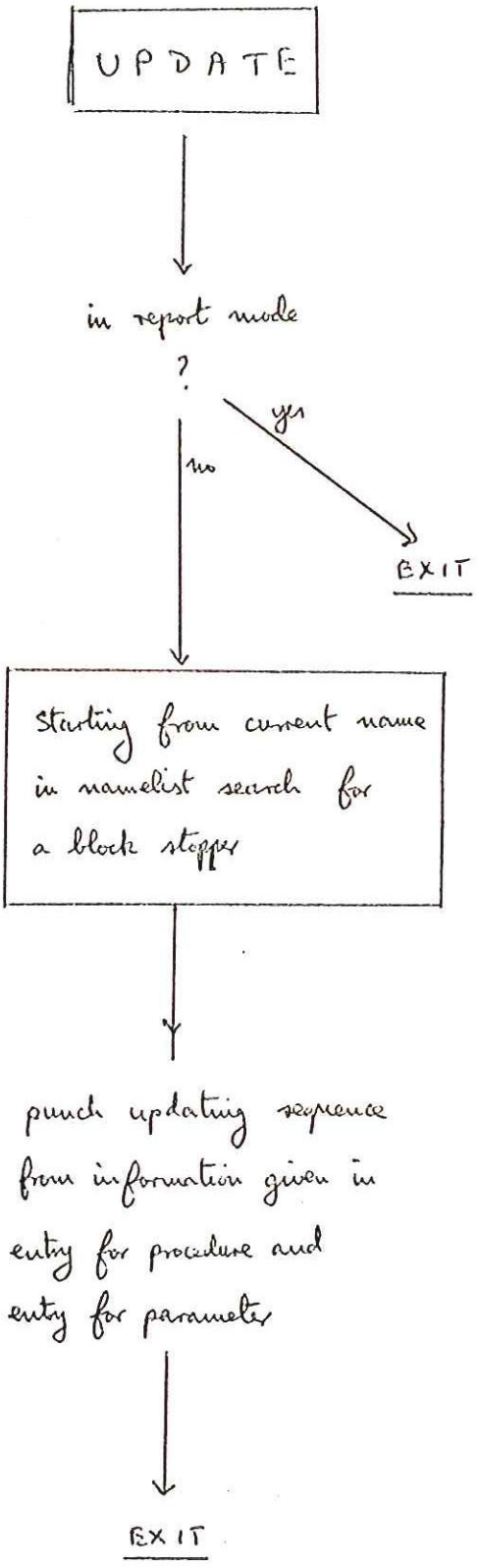





TAKE (P)
M ?
0
1
2
TAKID
EXIT
LASTDL ?
other
)
]
FAIL 30
UNSTAK (12)
EXIT
P=0 ?
yes
no
FAIL 30
EXIT
P=0 ?
yes
no
FAIL 31
CONS = 2 ?
yes
no
ARITH=0 ?
yes
no
FAIL 45
SECODL
CONS = 0 ?
no
yes
TYPBOX:=0
TYPBOX:=1
COMPIL
[TIC Q]
COMPIL
[TRC Q]
EXIT
EXIT
CALLED FROM ALMOST EVERYWHERE

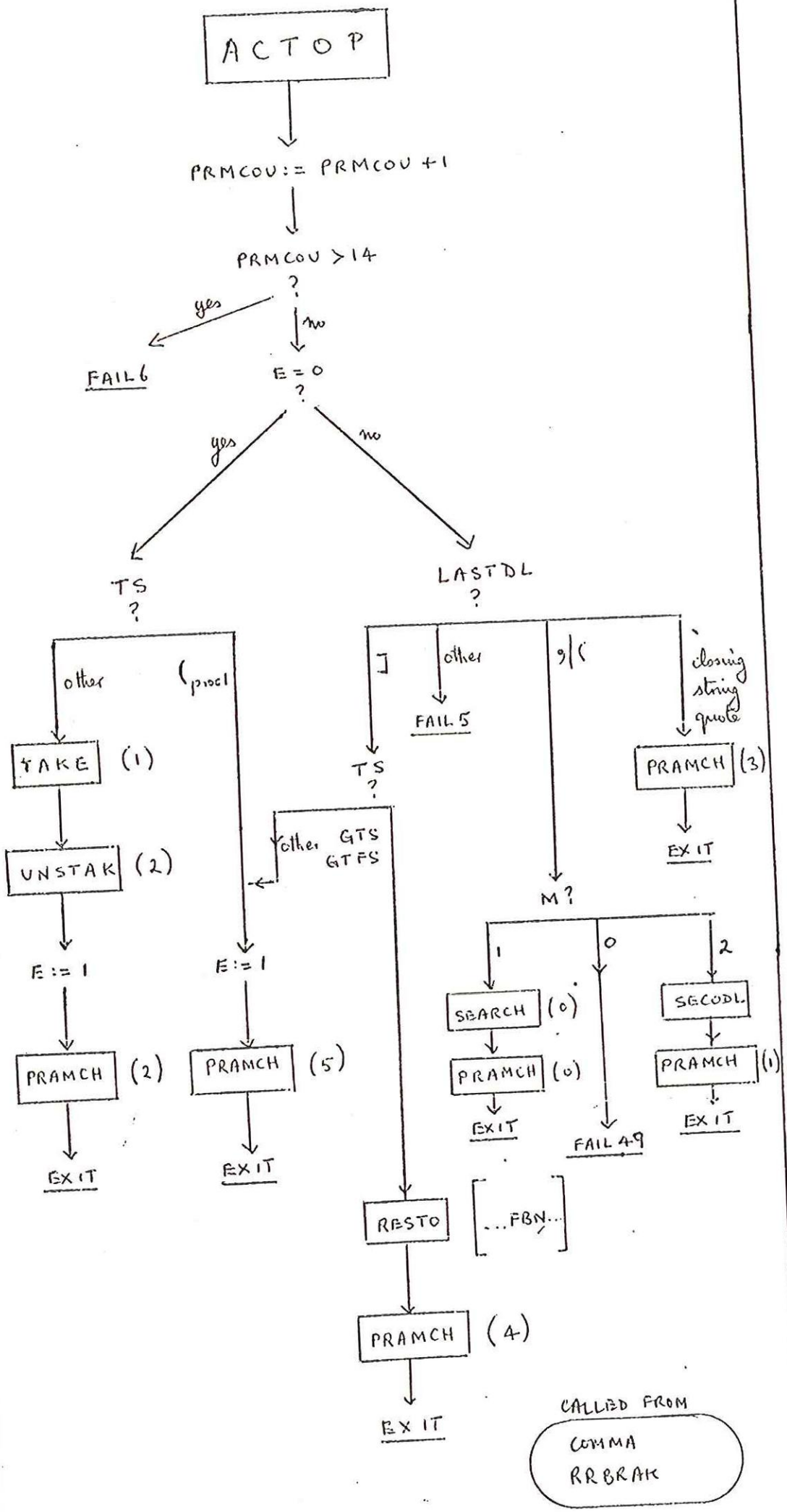


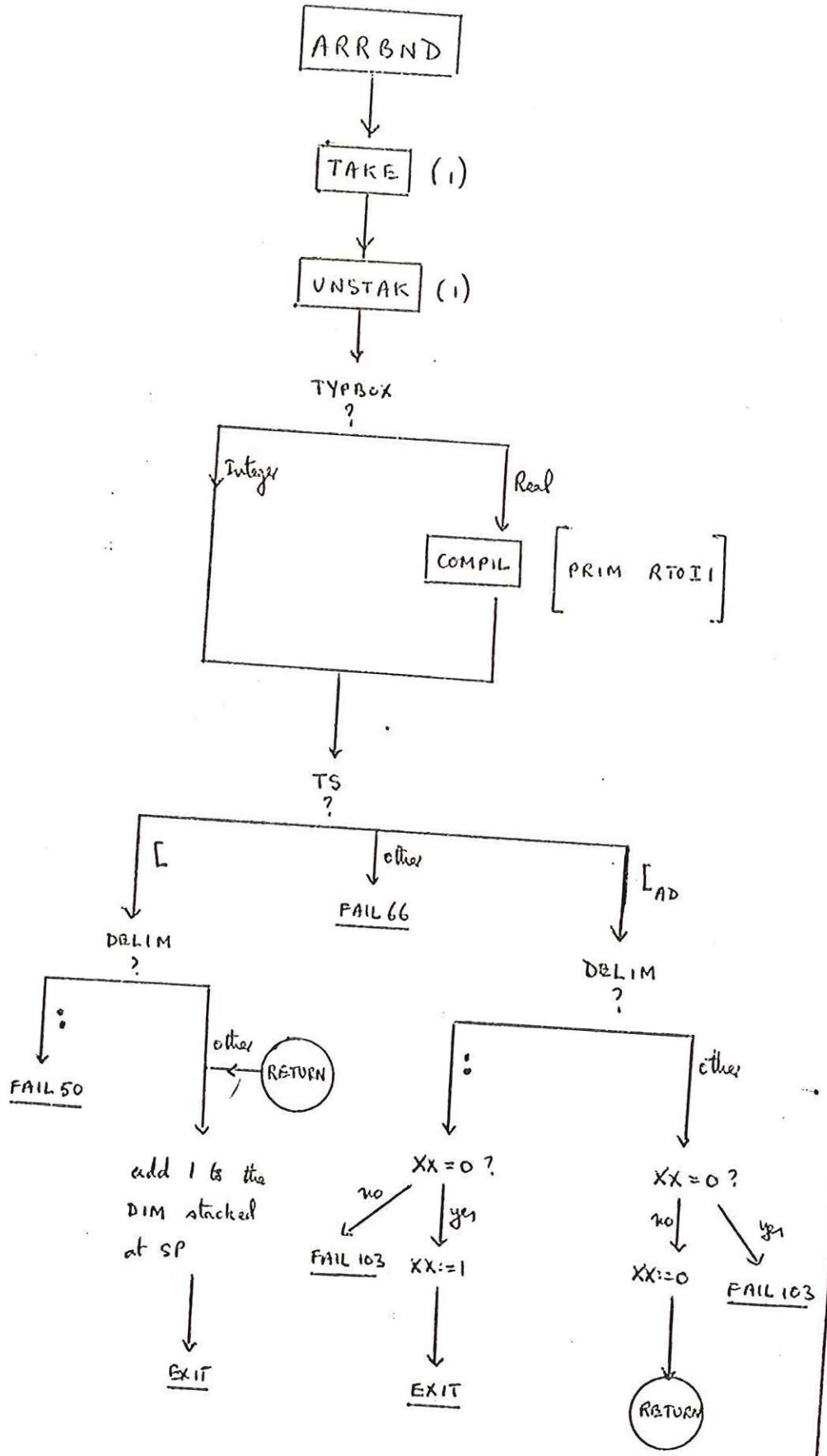
CALLED FROM
STEP FORCOM



CALLED FROM

RSBRAK
RRBRAK
FDMCOM


ACTOP
PRMCOU := PRMCOU + 1
PRMCOU > 14 ?
yes
no
FAIL 6
E = 0 ?
yes
no
TS ?
LASTDL ?
other
(proc
TAKE (1)
UNSTAK (2)
E := 1
PRAMCH (2)
EXIT
E := 1
PRAMCH (5)
EXIT
]
other
FAIL 5
TS ?
other, GTS GTFS
RESTO
[... FBN ...]
PRAMCH (4)
EXIT
9 (
closing string quote
PRAMCH (3)
EXIT
M ?
1
0
2
SEARCH (0)
PRAMCH (0)
EXIT
FAIL 49
SECODL
PRAMCH (1)
EXIT
CALLED FROM
COMMA
RRBRAK

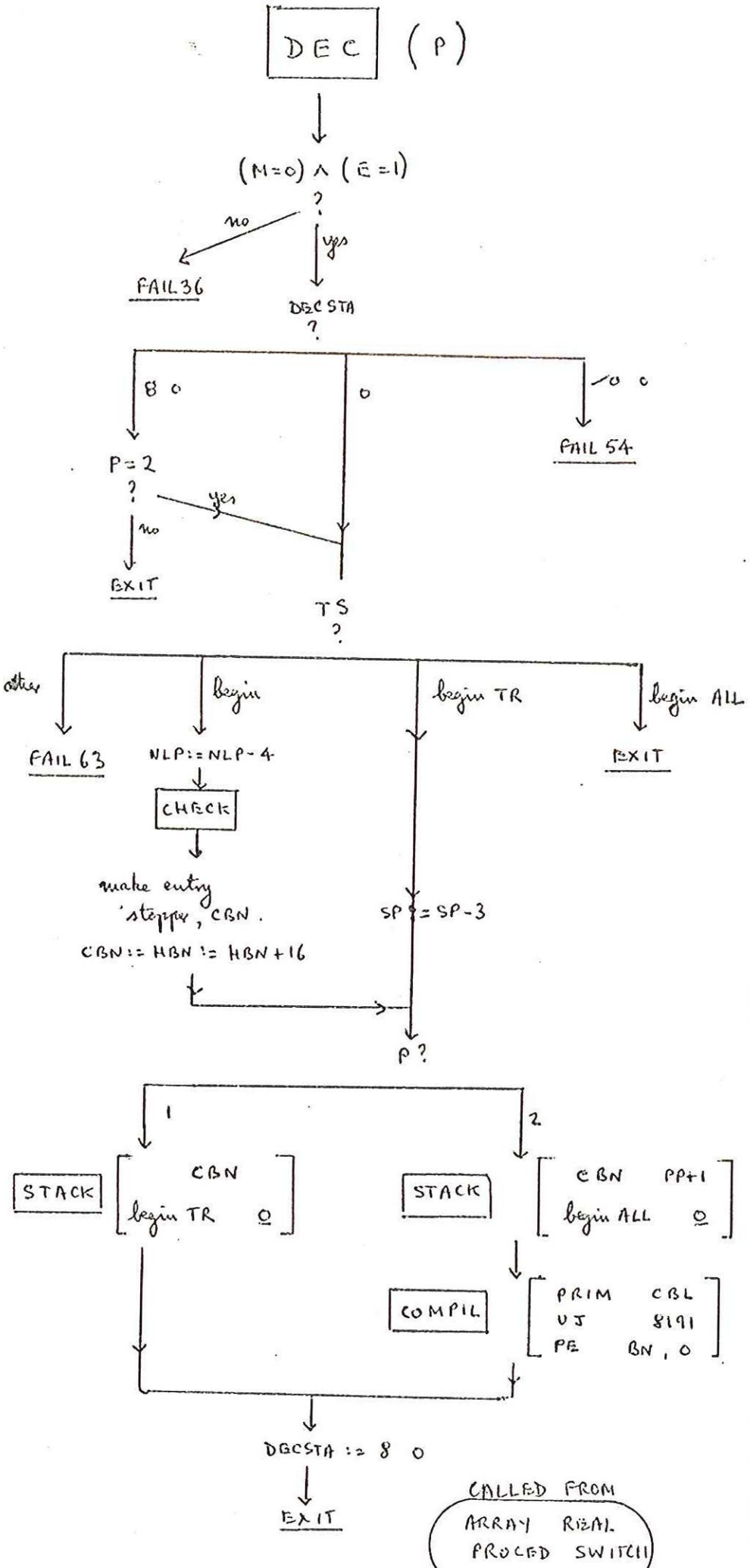


XX is the bit stored in the top of the stack

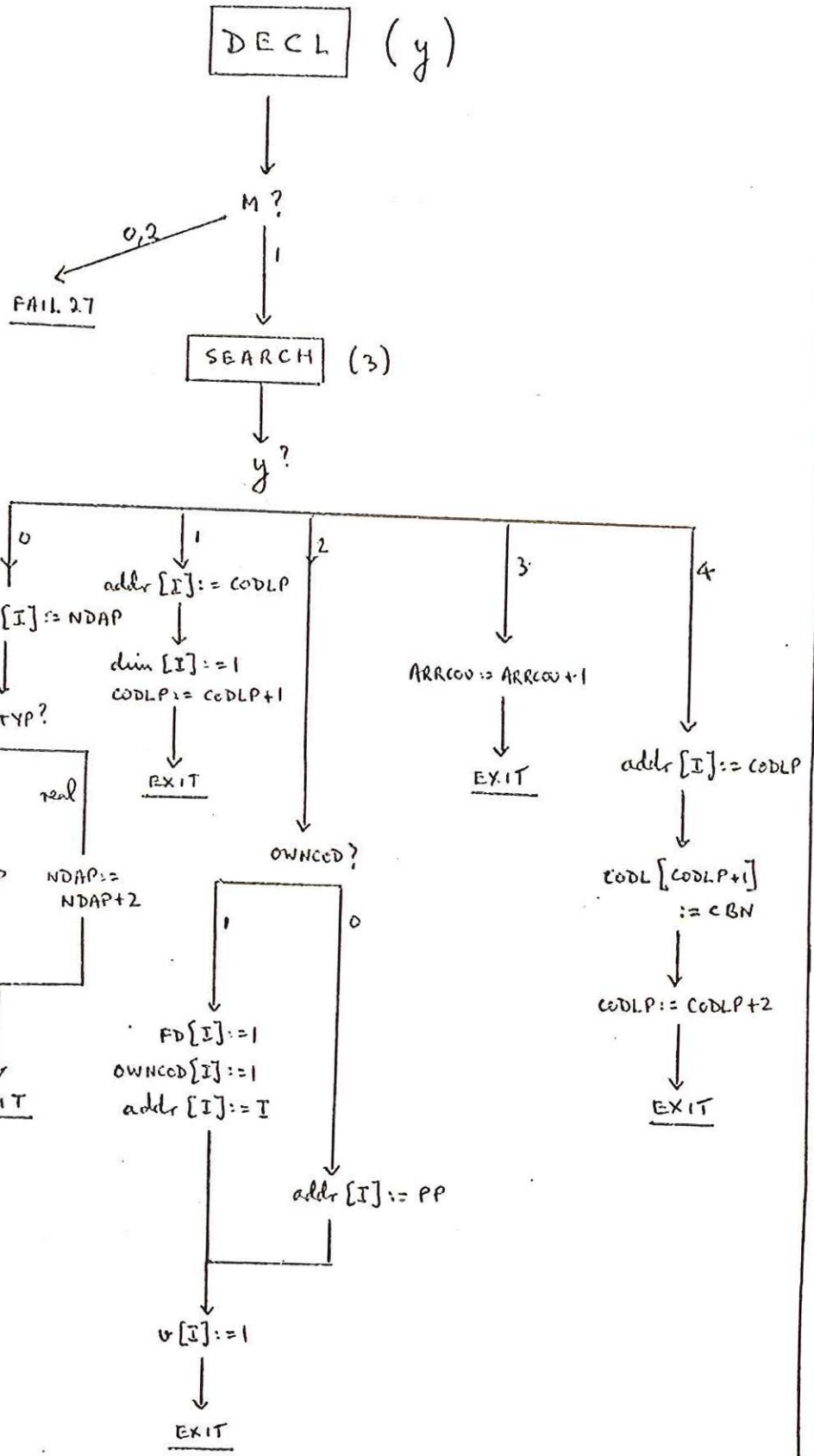
ARRBND

COLON

COMMA



Both these STACK operations begin by deleting Top item i.e entry at STACK+1



y is stored in W+1

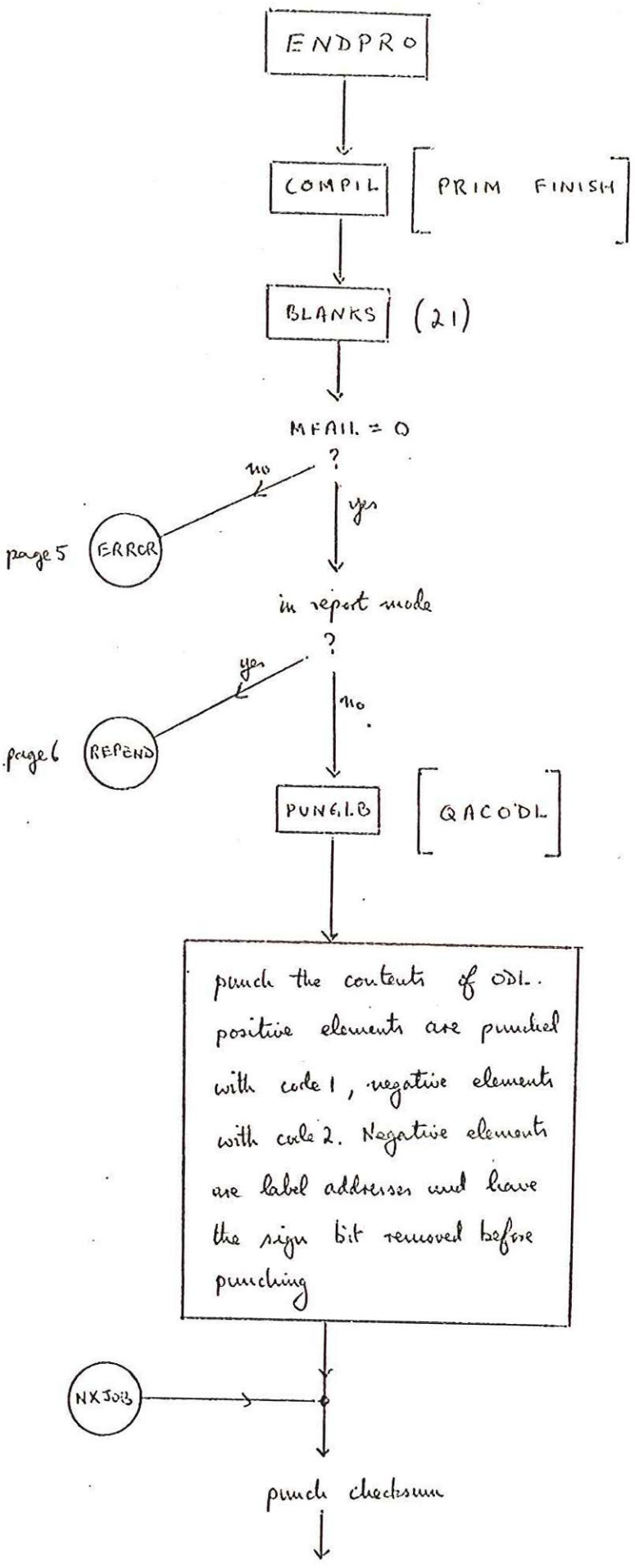
Make BN entry in CDL for a label

FD set to avoid fail in FCLAPS

v bit set to indicate within proc. body so as to fail recursive call

CALLED FROM

REAL PROCED
SWITCH SEMICO
LSBRAK COMMA FAIL

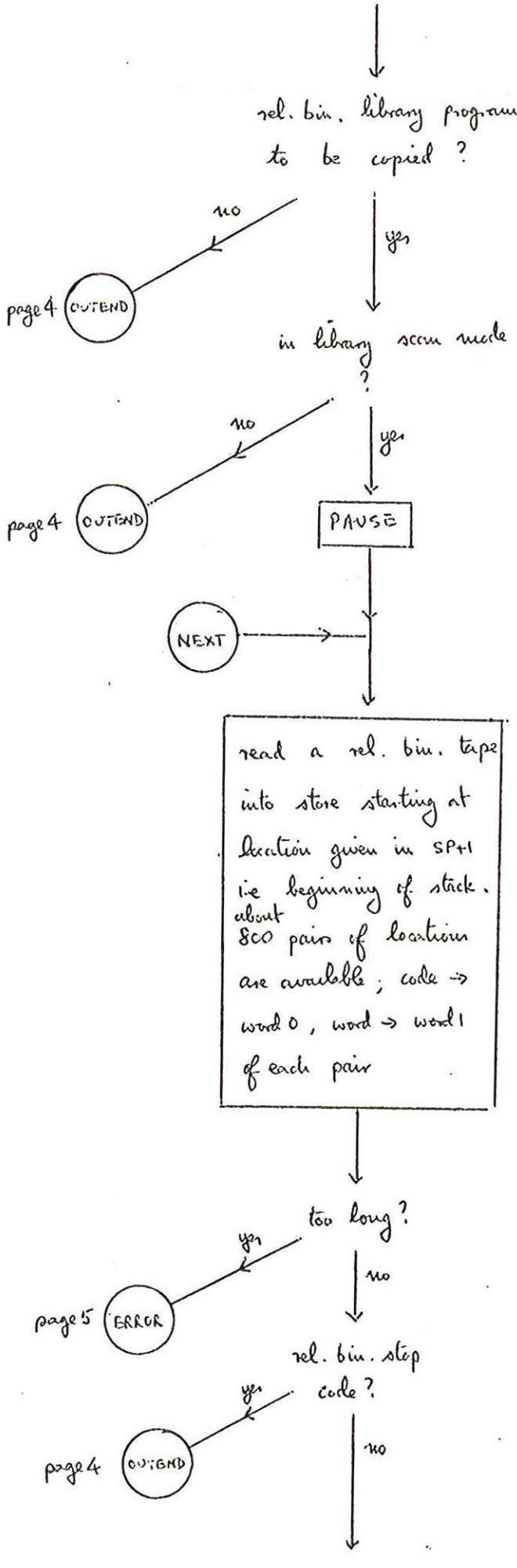


in report mode if OPTION 2 bit present

punch global L H. label

CALLED FROM
END RESTO
FAIL

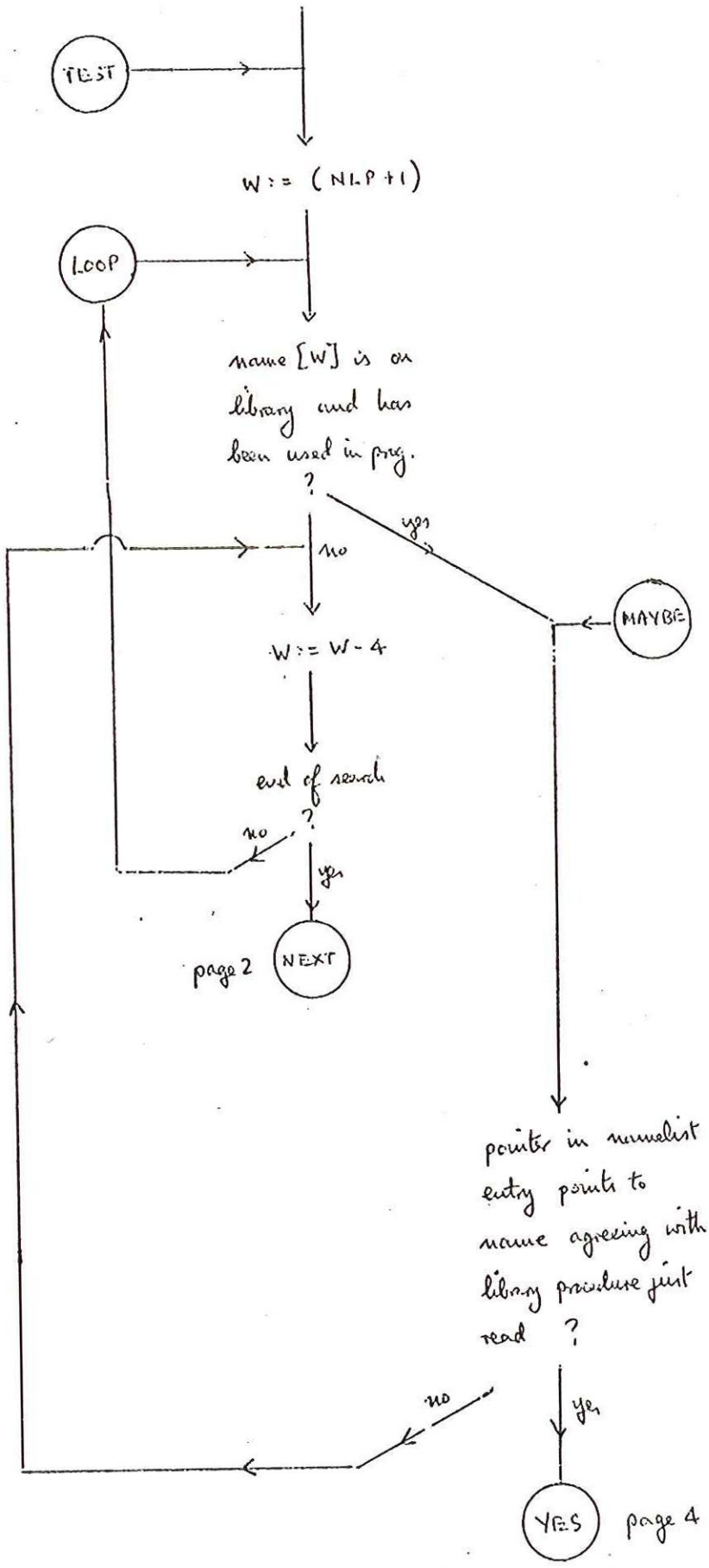
ENDPRO continued



in library scan mode if OPTION +8 bit present

Restart at 9 by User

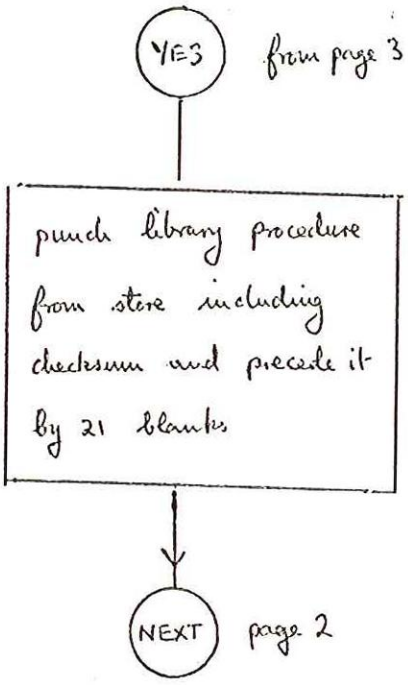
ENDPRO continued

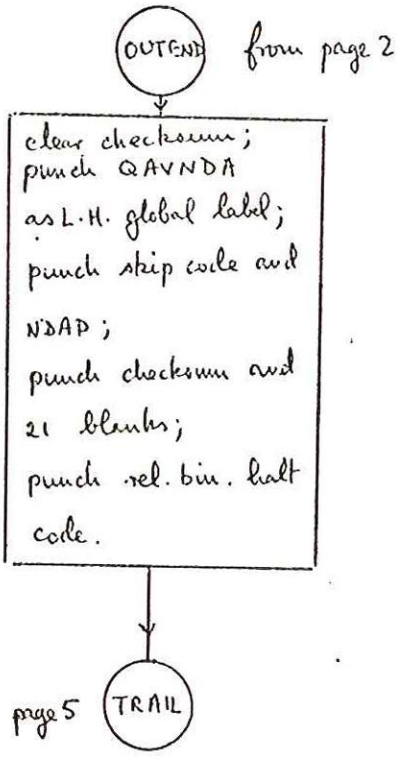


(NLP+1) points to top of namelist

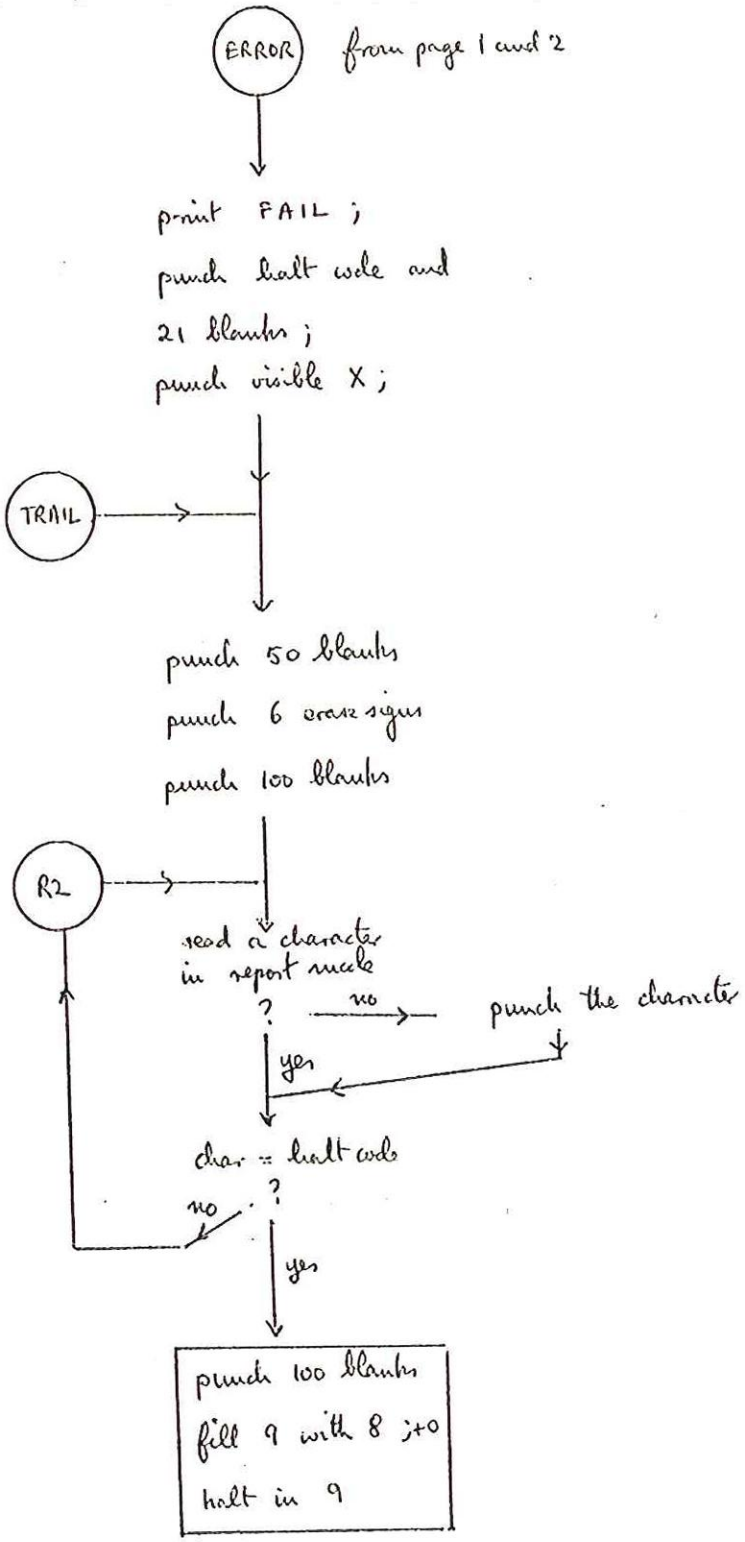
search ends at point including names declared in users outermost block

ENDPRO continued

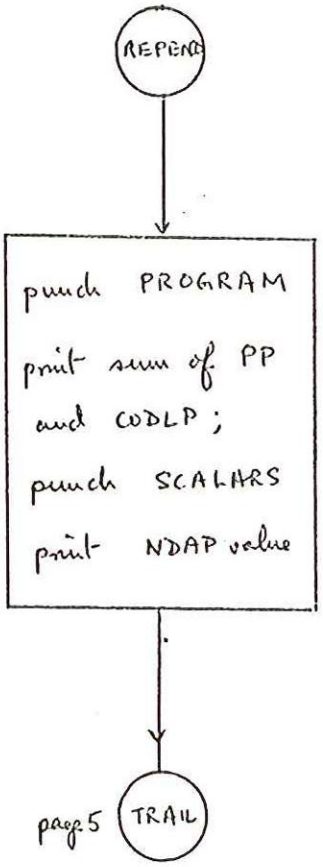


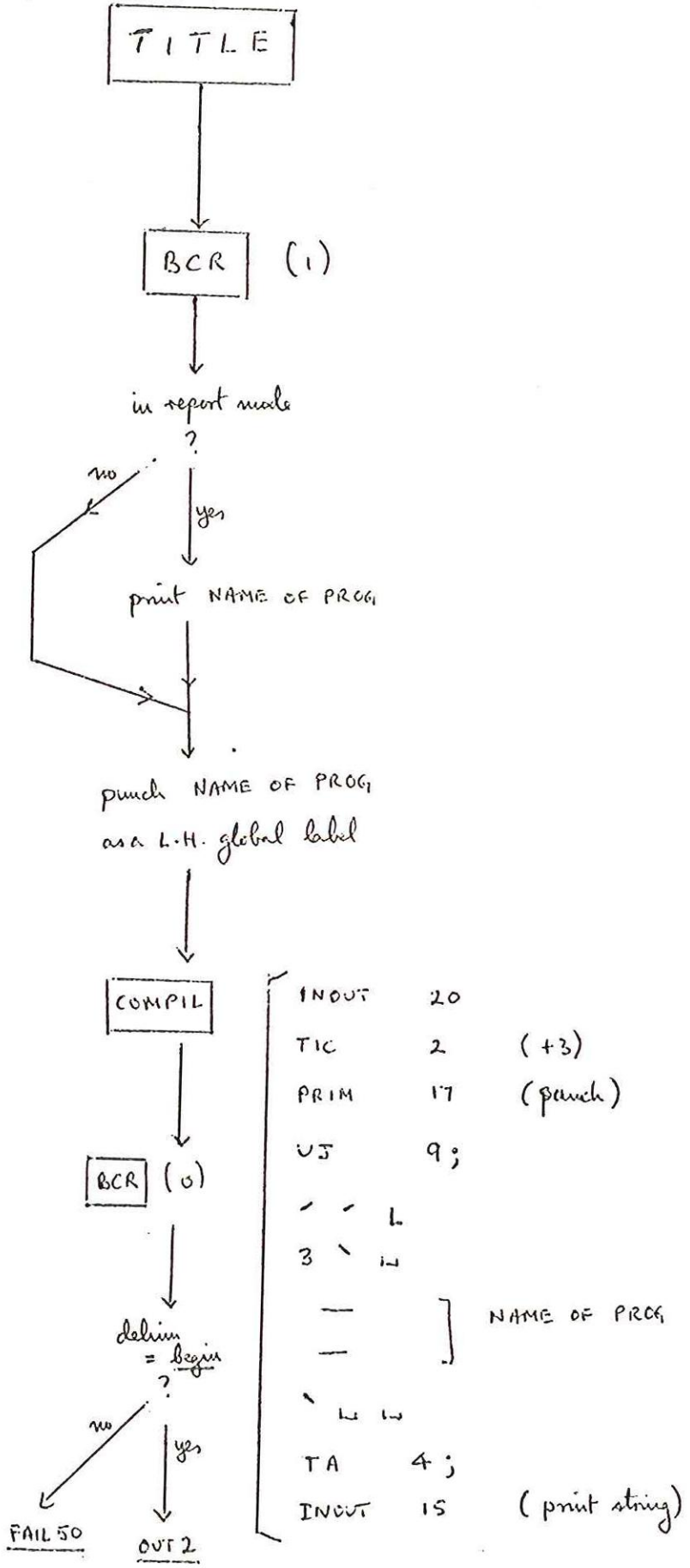


ENDPRO continued



ENDPRO continued

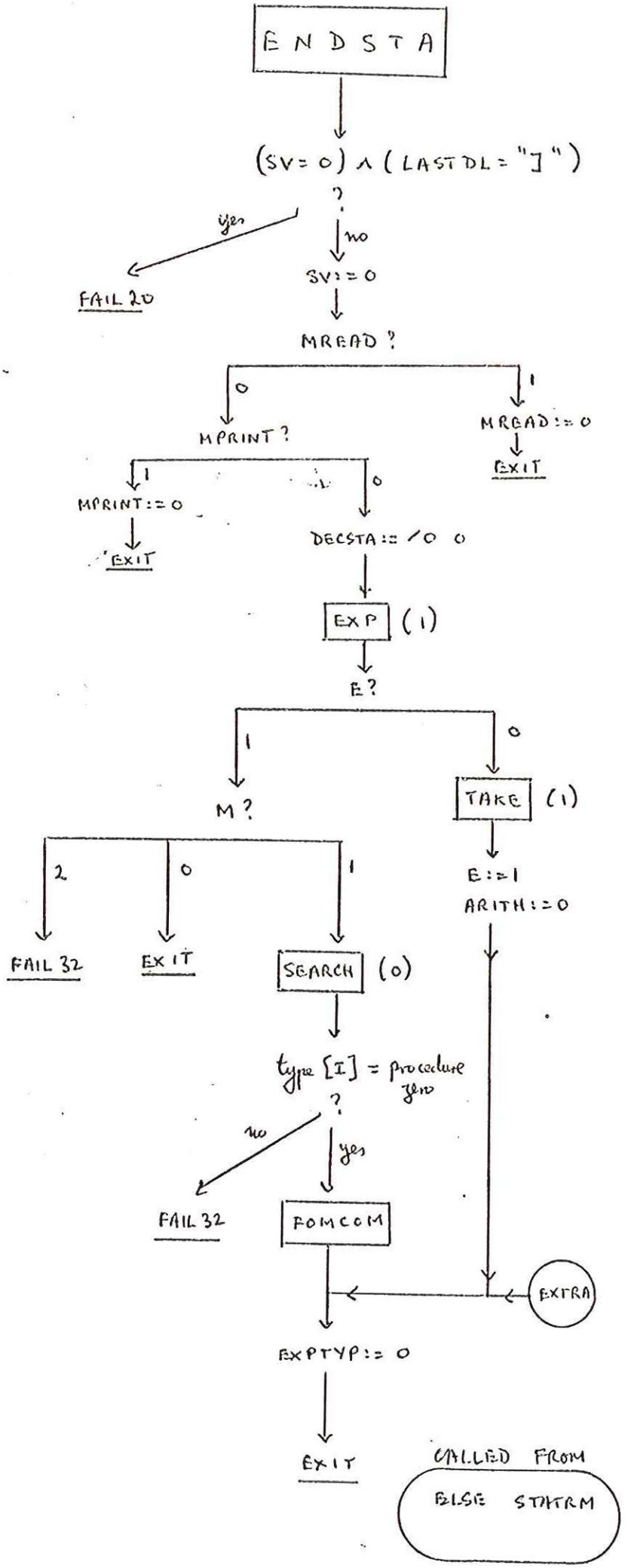


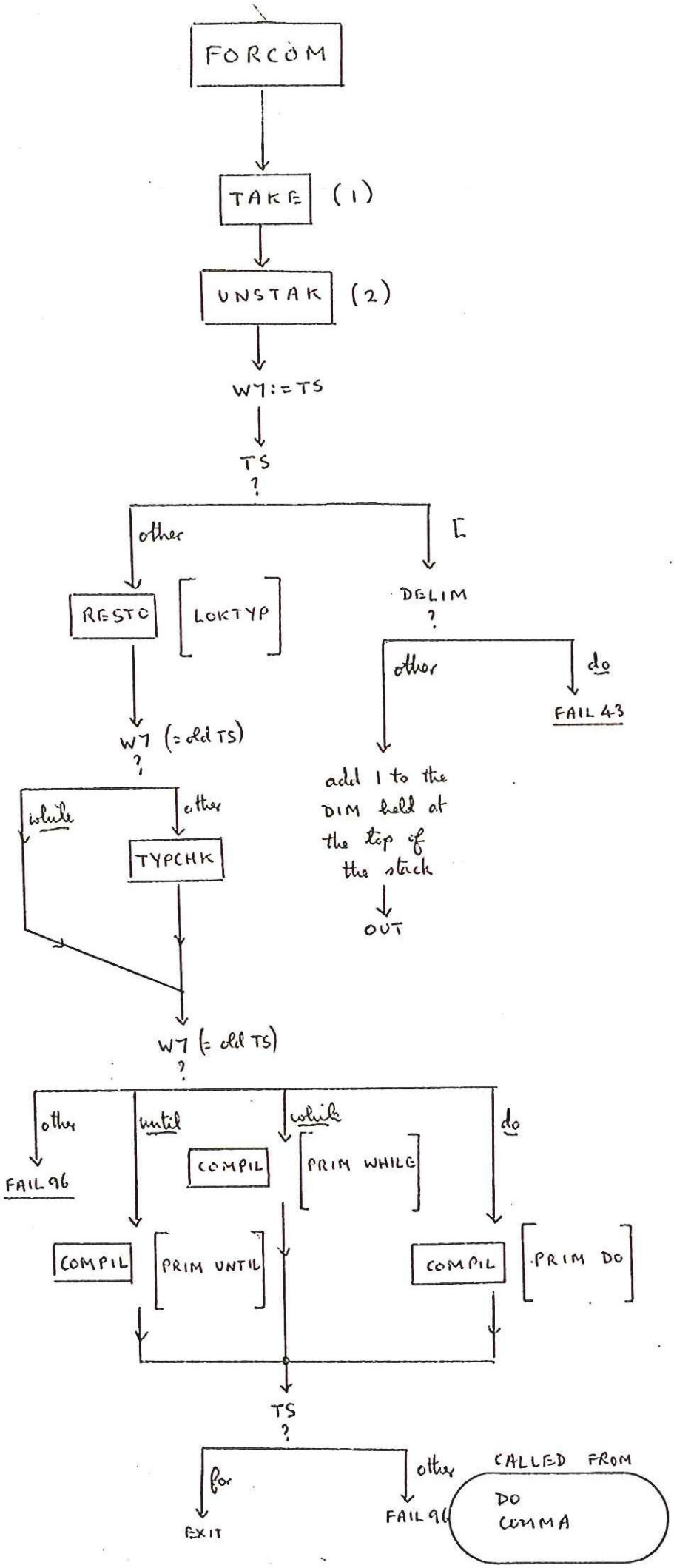


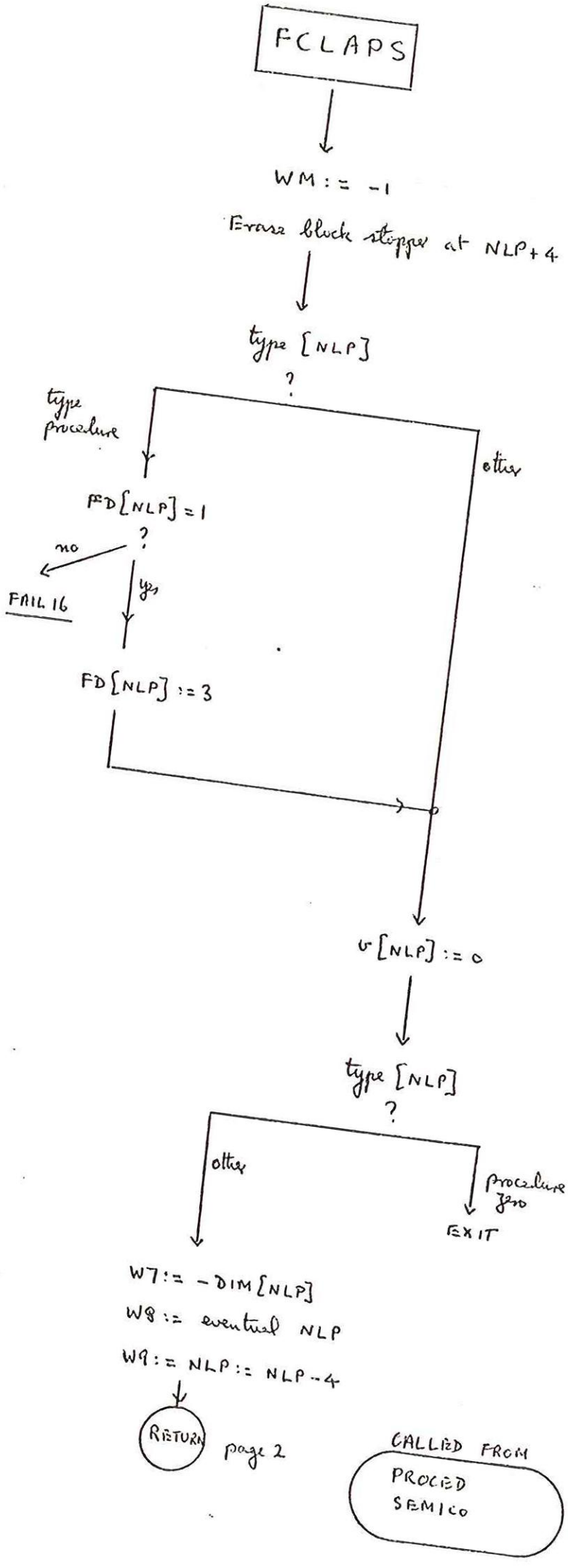
This section causes the name of the program to be output on punch(3) at the start of the program.

ENTERED FROM

START


ENDSTA
(SV = 0) ∧ (LASTDL = "]")
?
yes
no
FAIL 20
SV := 0
MREAD ?
0
1
MPRINT ?
MREAD := 0
EXIT
1
0
MPRINT := 0
EXIT
DECSTA := /0 0
EXP (1)
E?
1
0
M?
TAKE (1)
2
0
1
E := 1
ARITH := 0
FAIL 32
EXIT
SEARCH (0)
type [I] = procedure
?
no
yes
FAIL 32
FOMCOM
EXTRA
EXPTYP := 0
EXIT
CALLED FROM
ELSE STATRM

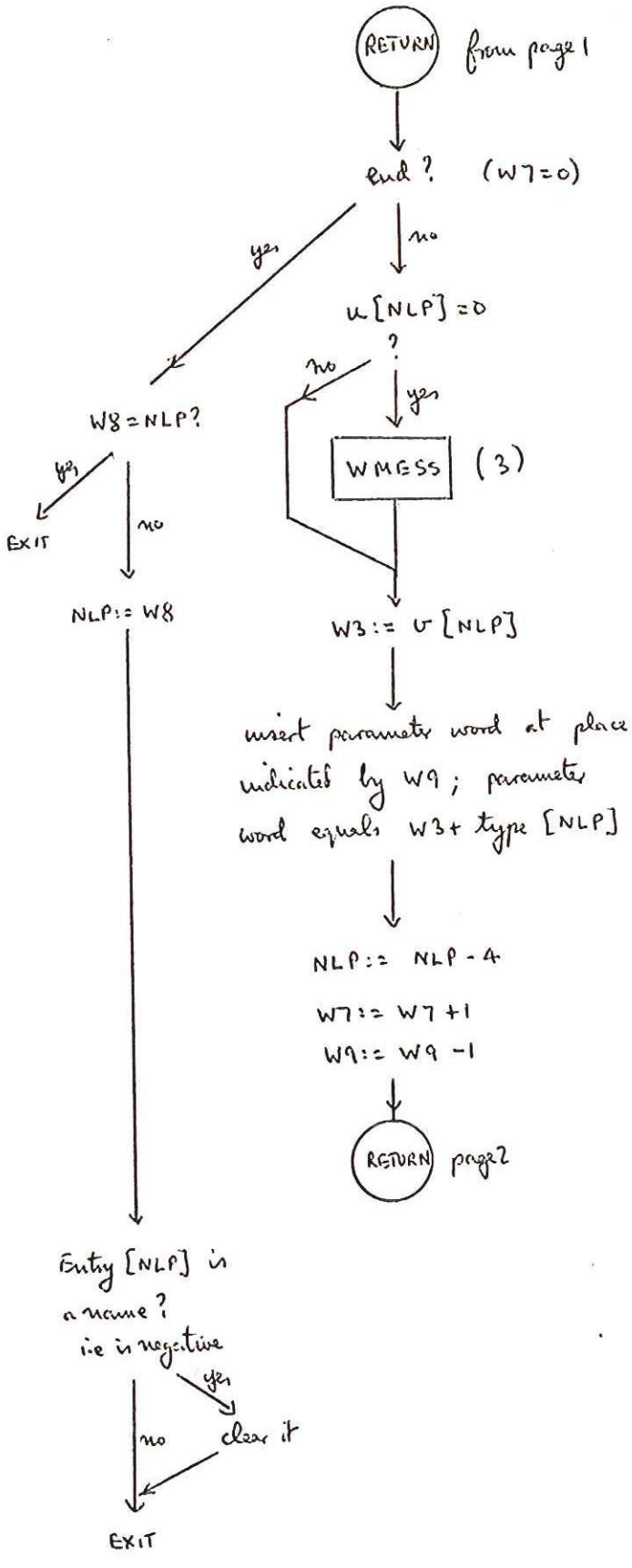

FORCOM
TAKE (1)
UNSTAK (2)
W7 := TS
TS ?
other
[
RESTO
[LOKTYP
DELIM ?
other
do
FAIL 43
W7 (= old TS) ?
while
other
TYPCHK
add 1 to the DIM held at the top of the stack
OUT
W7 (= old TS) ?
other
until
while
do
FAIL 96
COMPIL
PRIM WHILE
COMPIL
PRIM UNTIL
COMPIL
PRIM DO
TS ?
for
other
EXIT
FAIL 96
CALLED FROM
DO
COMMA



v bit = 1 during proc. body - used to detect recursion

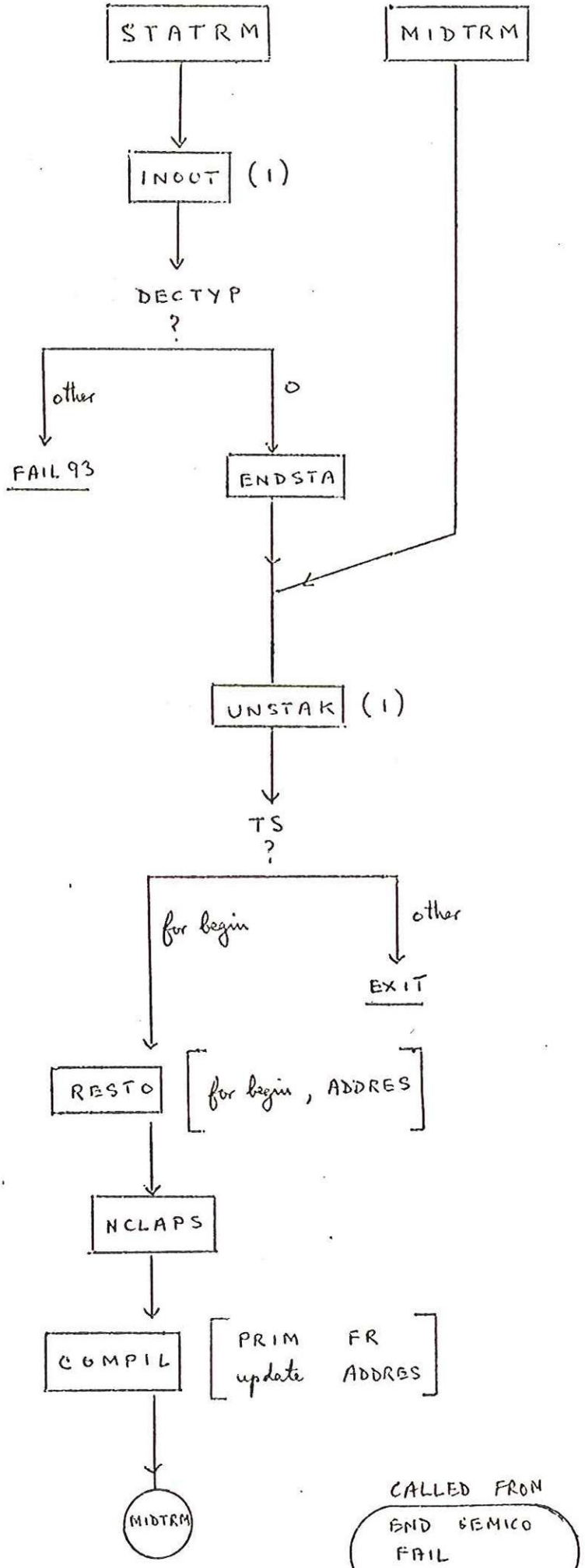
W8 := NLP - 4*((DIM+3) div 4)

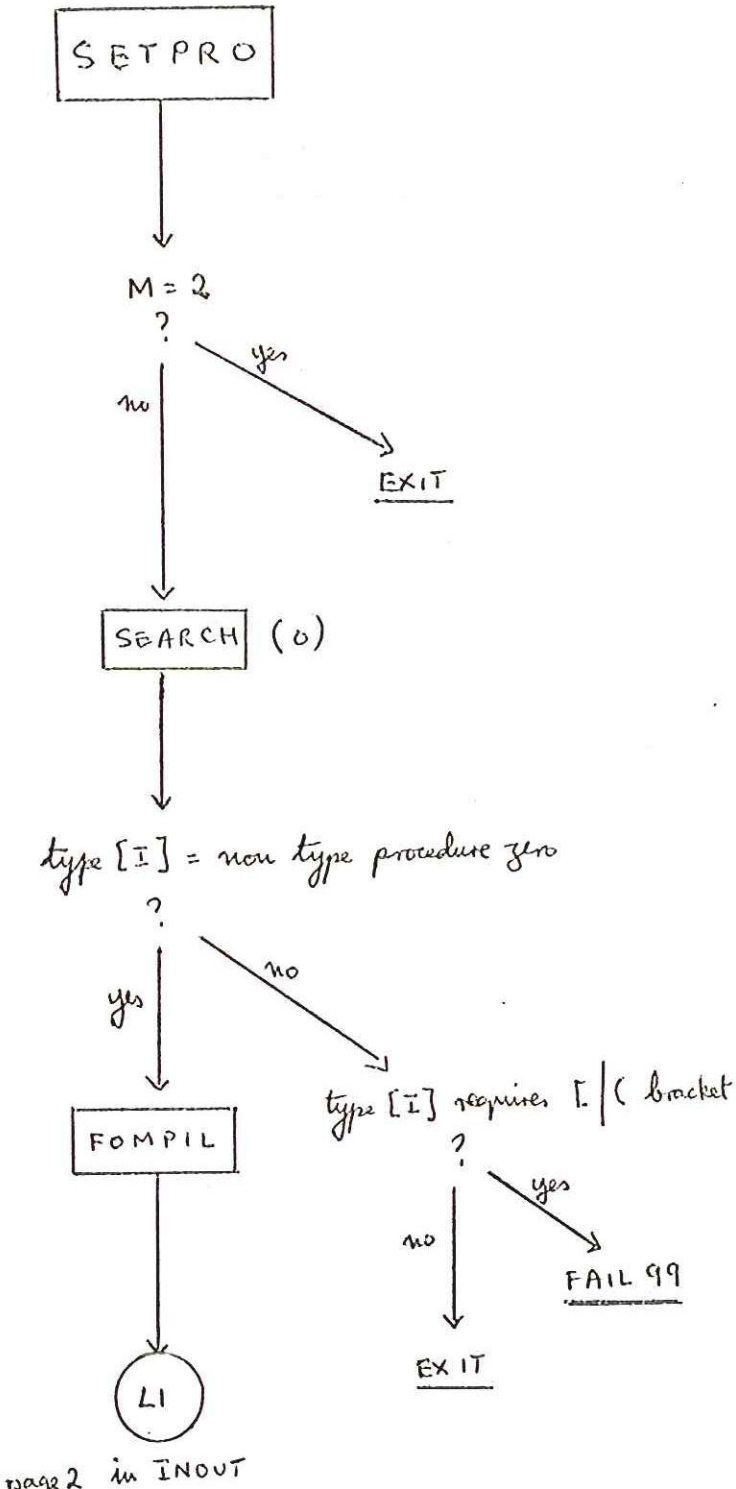
FCLAPS continued



The first parameter word is adjacent to the procedure name.

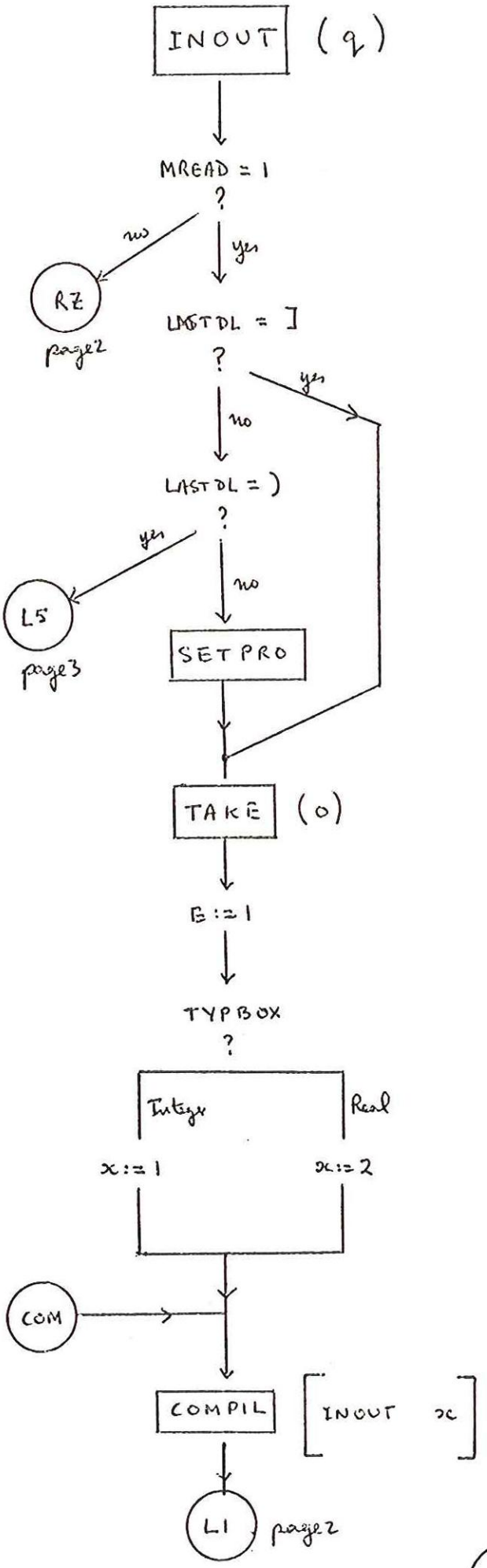
There may be a vestige of a name occupying word 0 of the parameter word group of 4.


STATRM
MIDTRM
INOUT (1)
DECTYP ?
other
0
FAIL 93
ENDSTA
UNSTAK (1)
TS ?
for begin
other
EXIT
RESTO
[for begin , ADDRES]
NCLAPS
COMPIL
[PRIM FR
update ADDRES]
MIDTRM
CALLED FROM
END GEMICO
FAIL



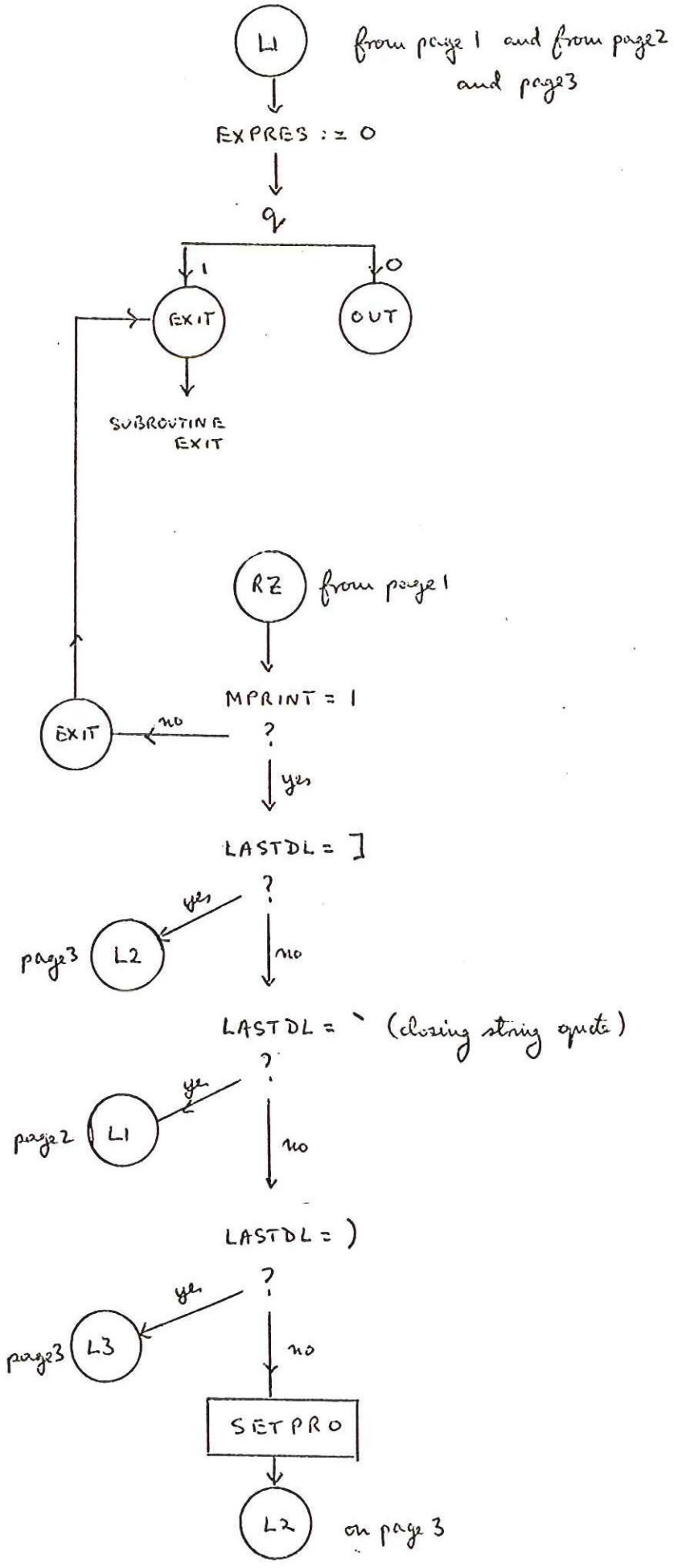
CALLED FROM INOUT

q is stored in W+11

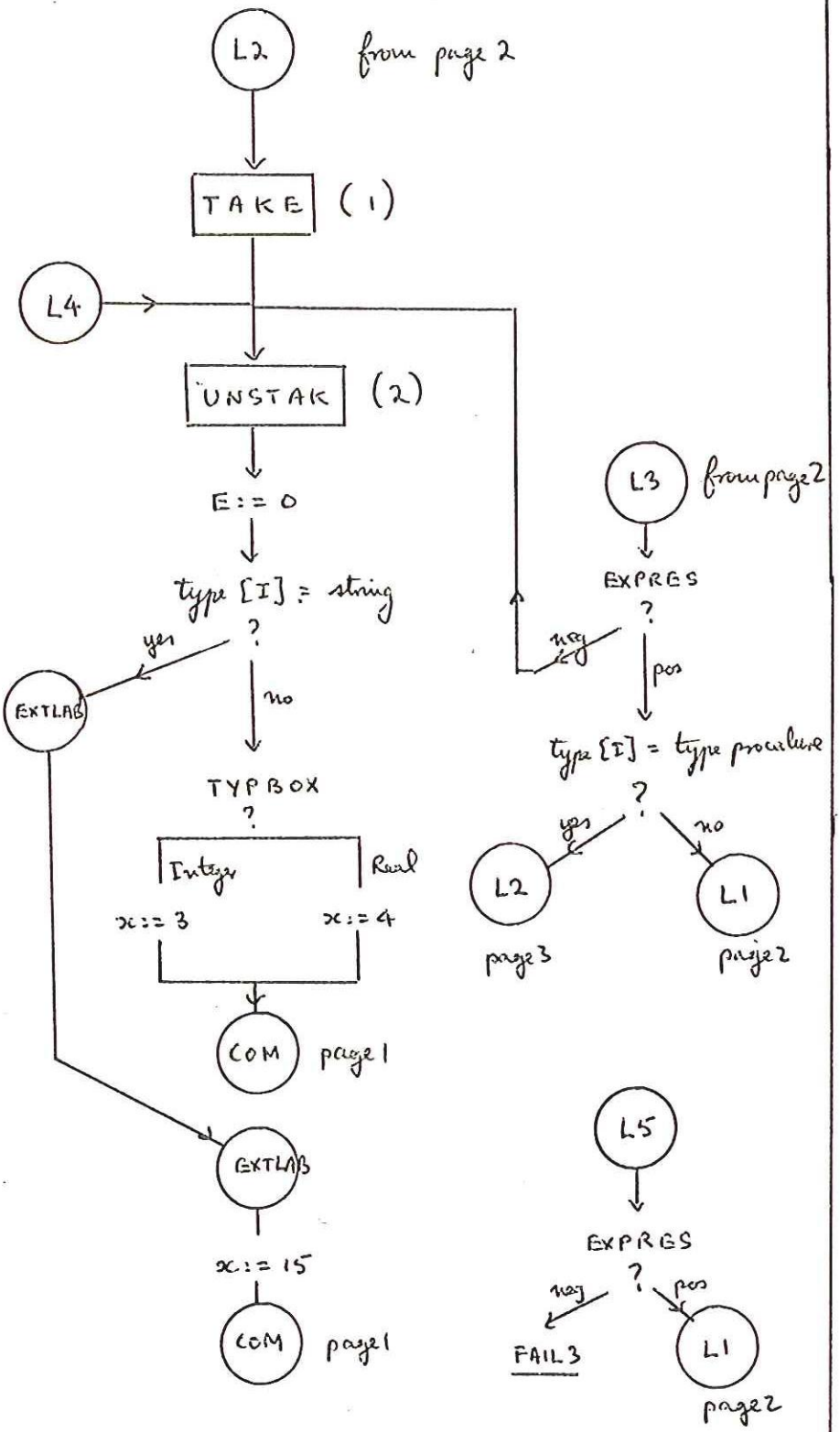


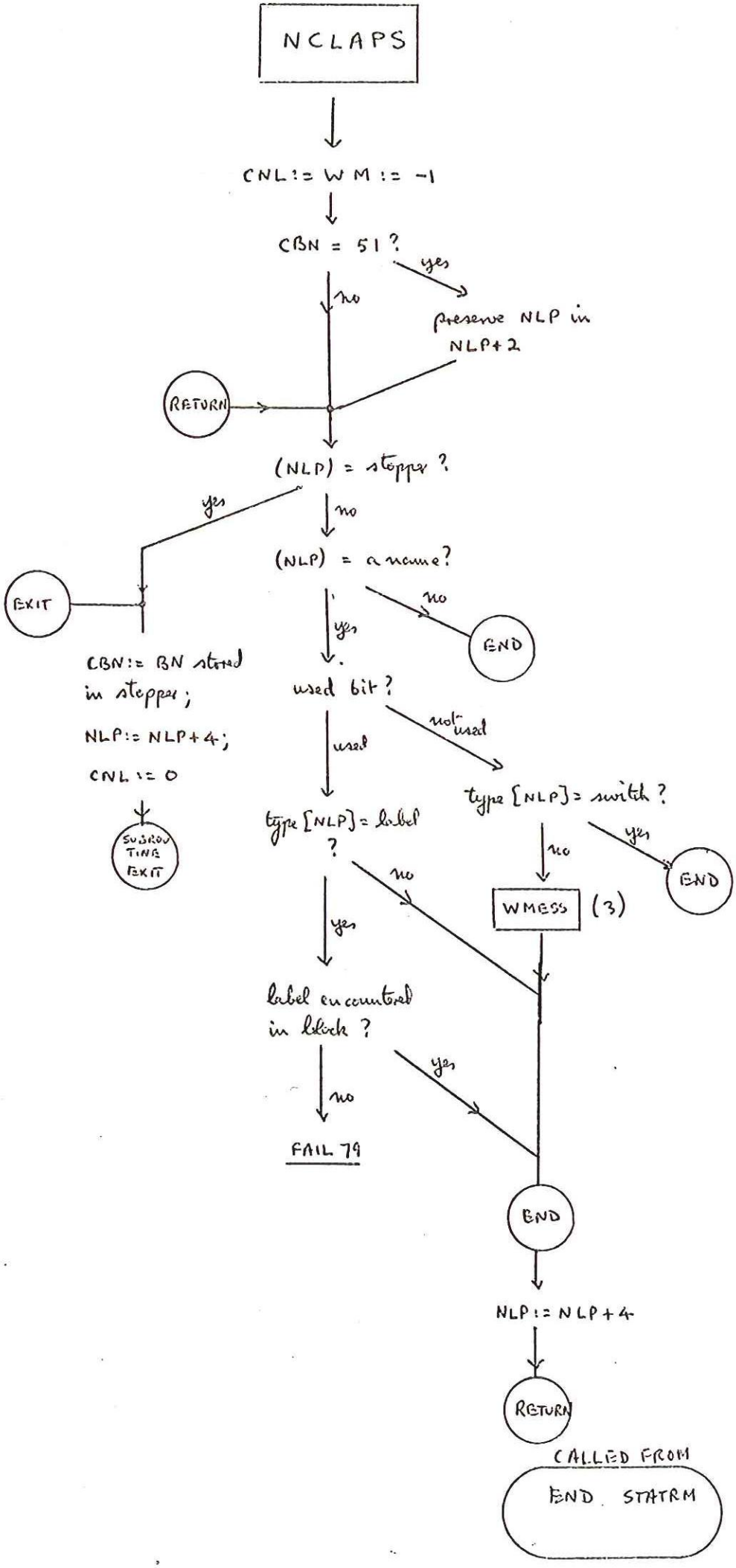
CALLED FROM
ELSE COMMA
STATRM

INOUT (q) continued.



INOUT (9) continued

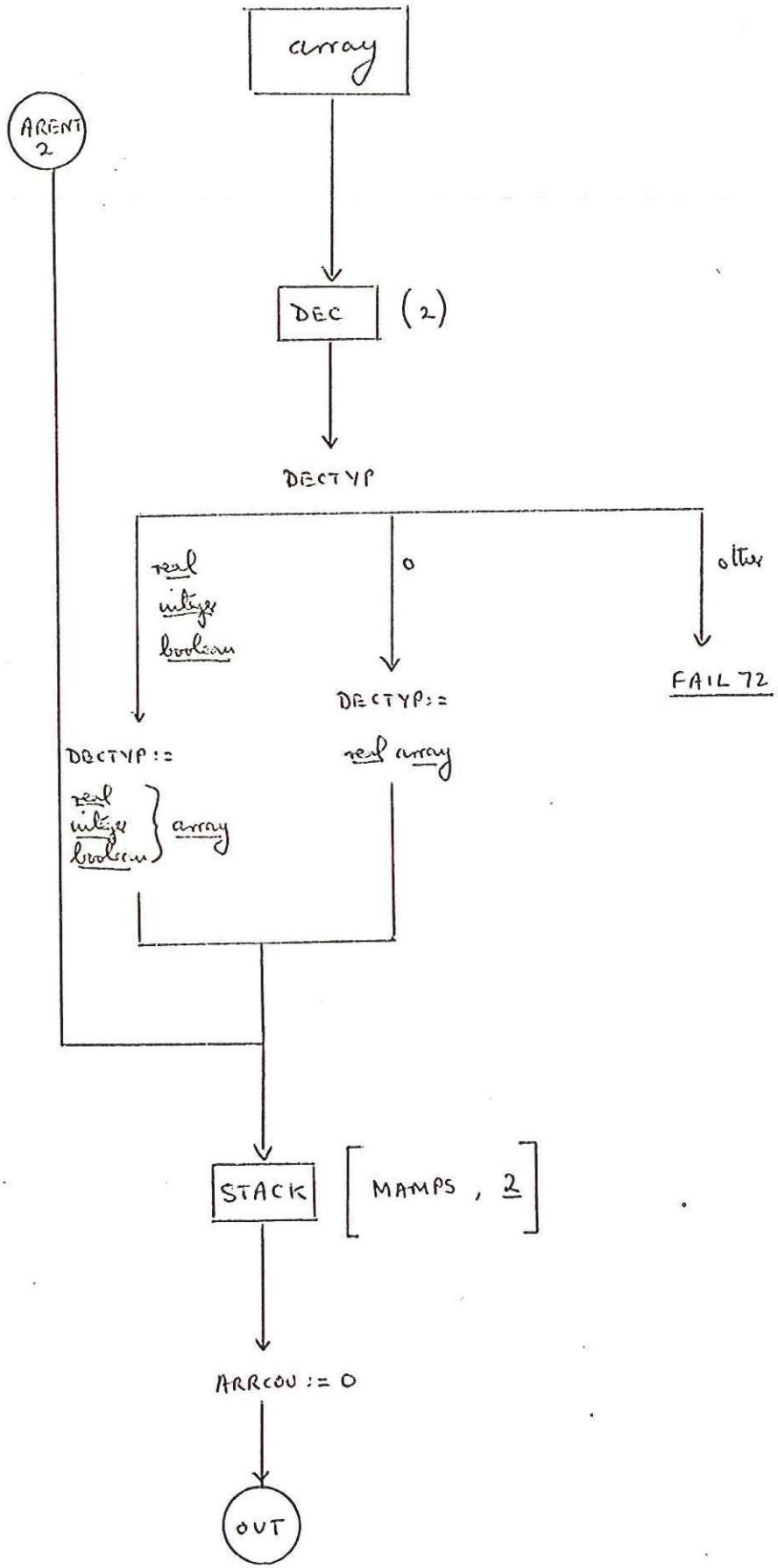




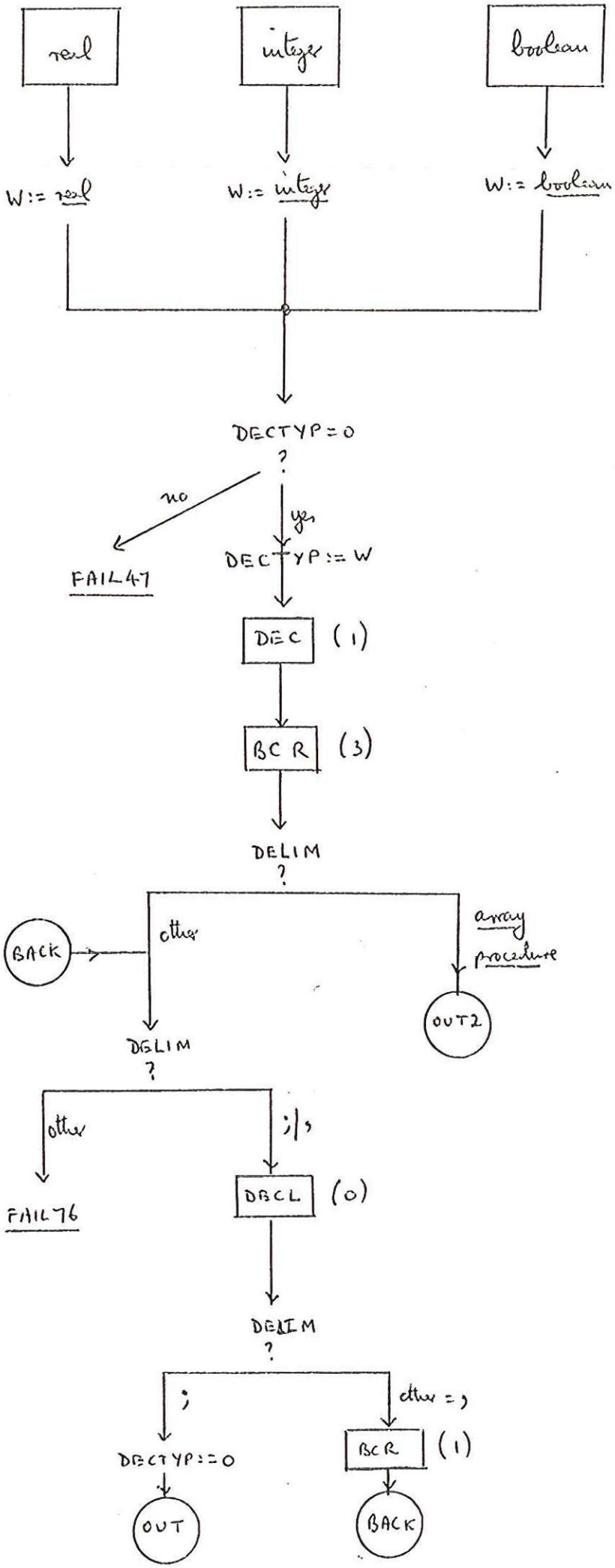
preservation purpose is so that library scan can include
cate
algol
} procedures
stopper is -1

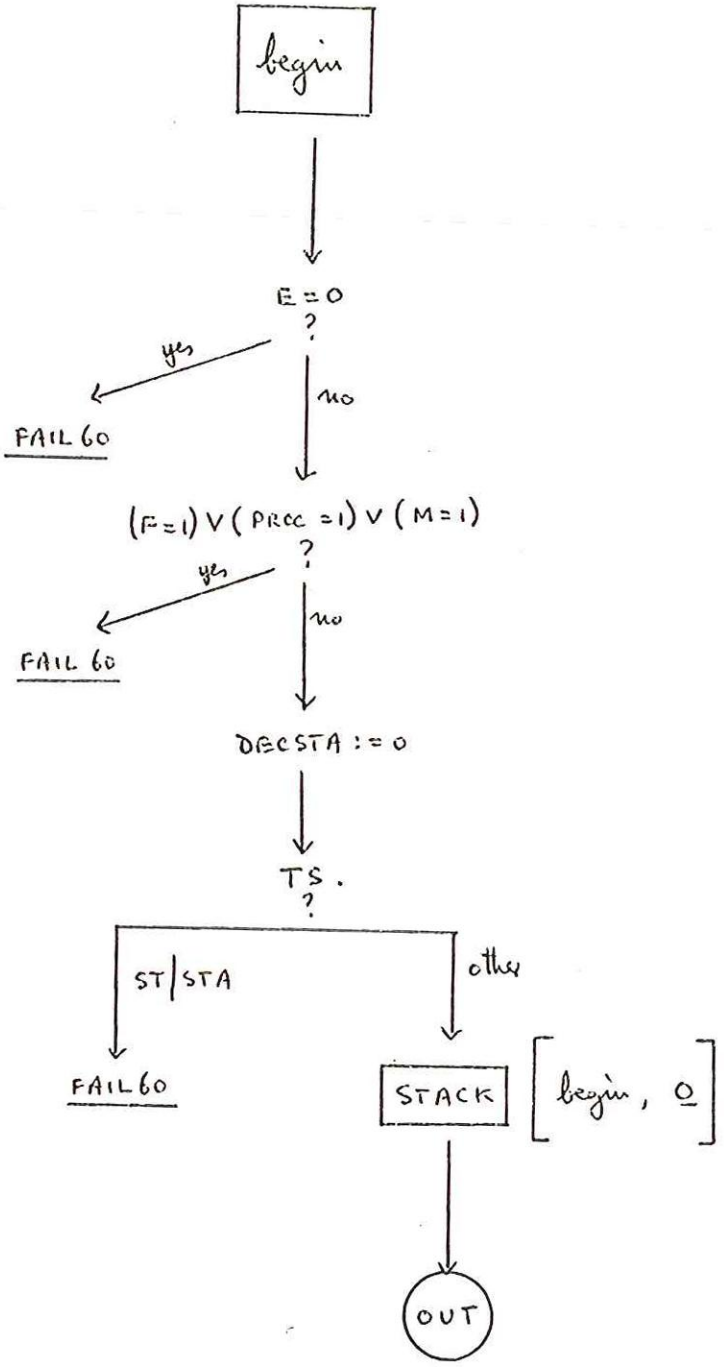
(NLP) is negative for a name

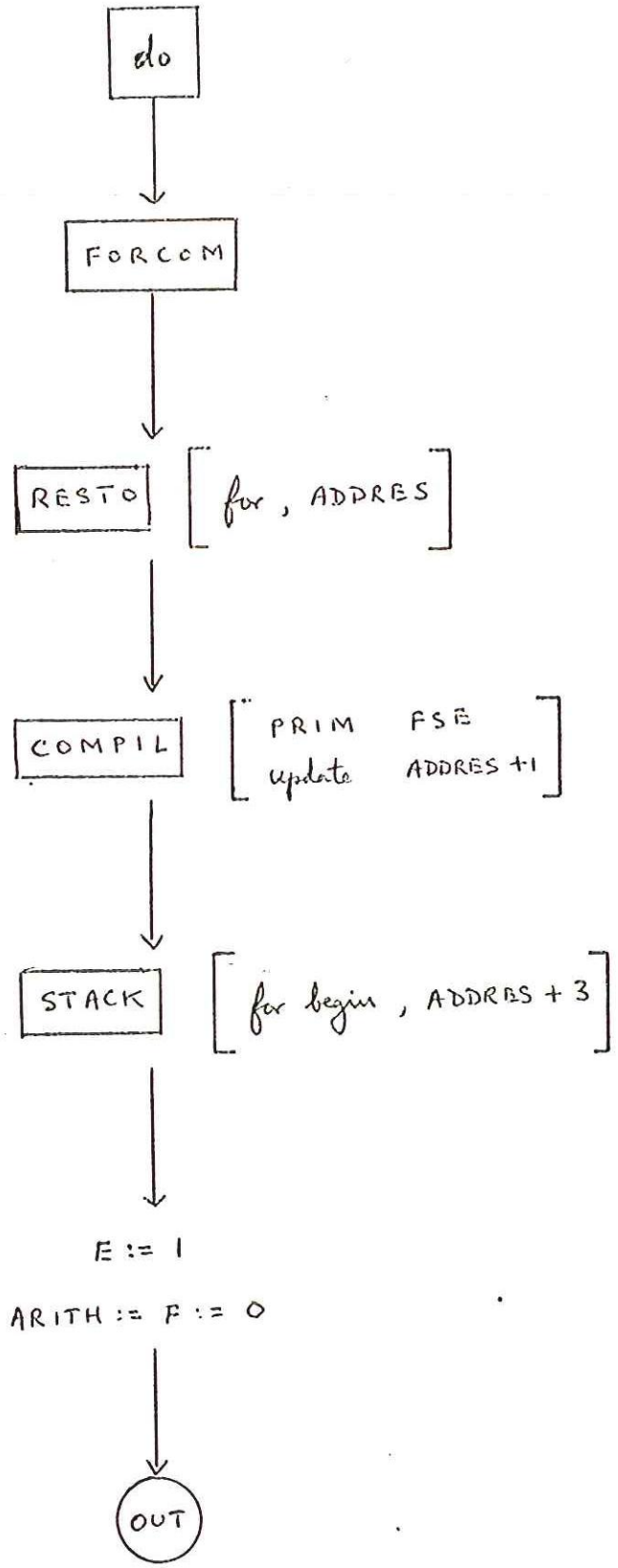
examine CODL entry: non zero if encountered.

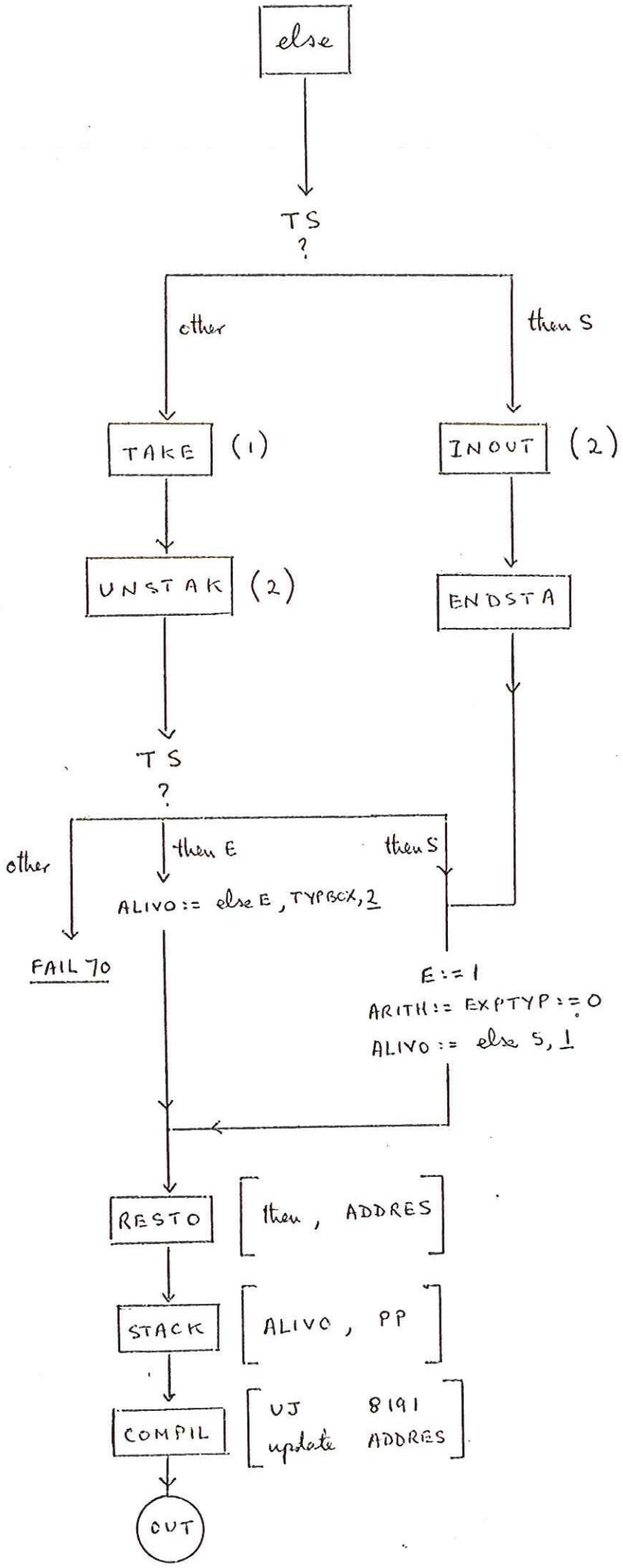


ARGNT 2 is an entry from RSBRAK.

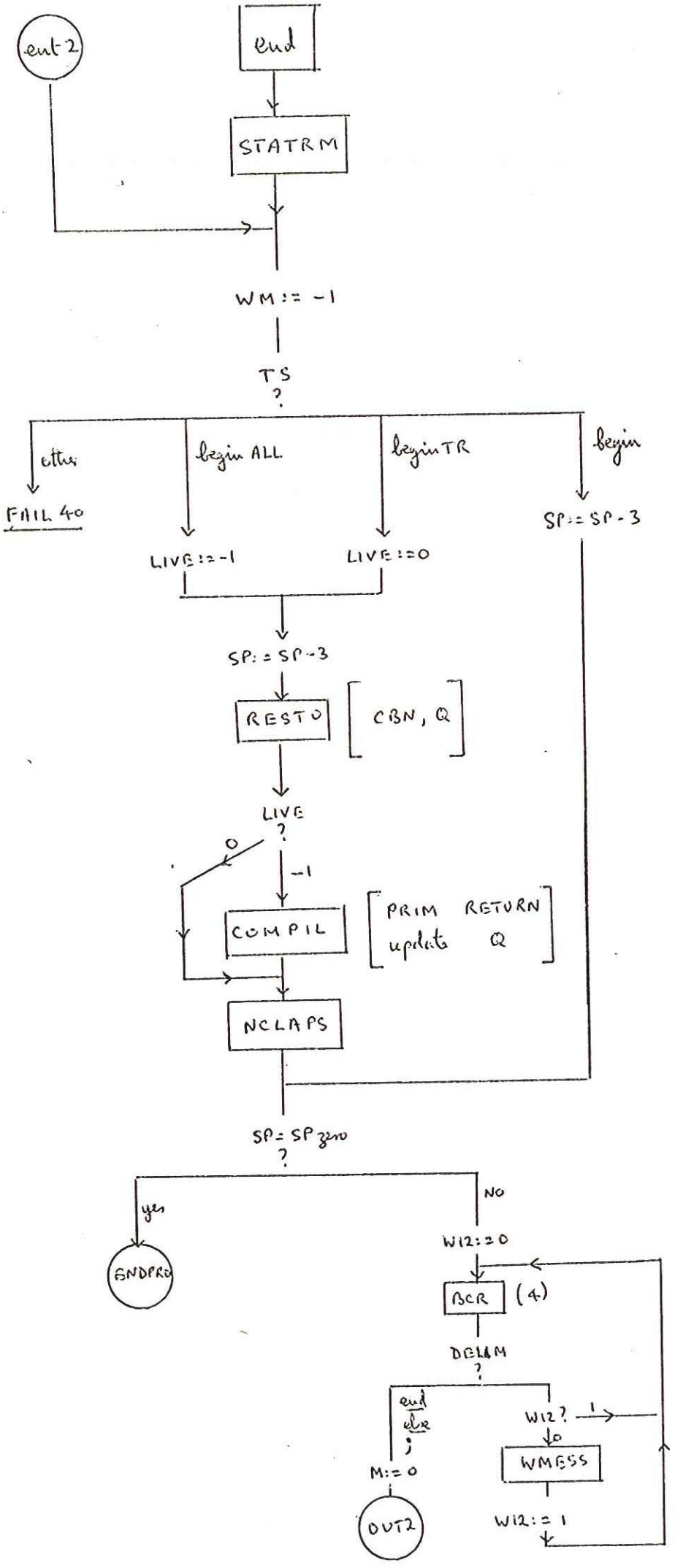

real
integer
boolean
W:= real
W:= integer
W:= boolean
DECTYP=0
?
no
yes
FAIL47
DECTYP:= W
DEC
(1)
BCR
(3)
DELIM
?
BACK
other
array
procedure
OUT2
DELIM
?
other
;|,
FAIL76
DBCL
(0)
DELIM
?
;
other =,
DECTYP:=0
BCR
(1)
OUT
BACK

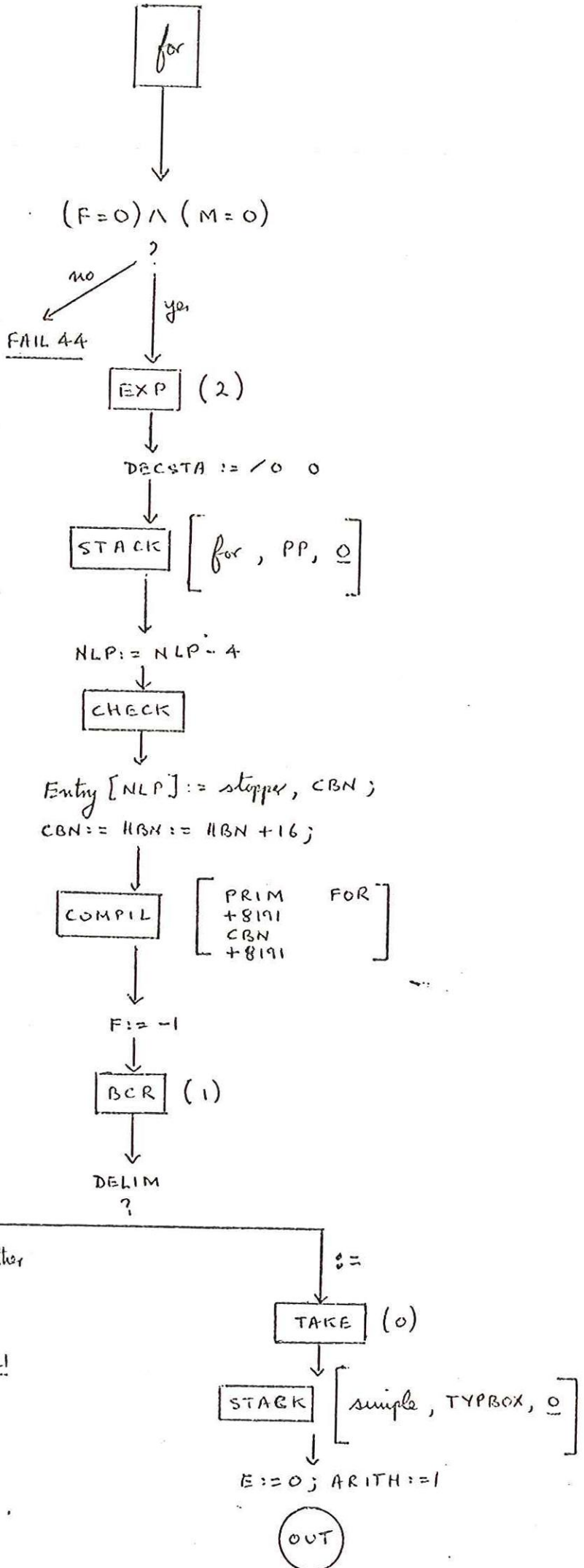

begin
E=0
?
yes
no
FAIL 60
(F=1) V (PROC=1) V (M=1)
?
yes
no
FAIL 60
DECSTA := 0
TS.
?
ST/STA
other
FAIL 60
STACK
[begin , 0]
OUT


do
FORCOM
RESTO
[for , ADDRES]
COMPIL
[PRIM FSE
update ADDRES +1]
STACK
[for begin , ADDRES + 3]
E := 1
ARITH := F := 0
OUT

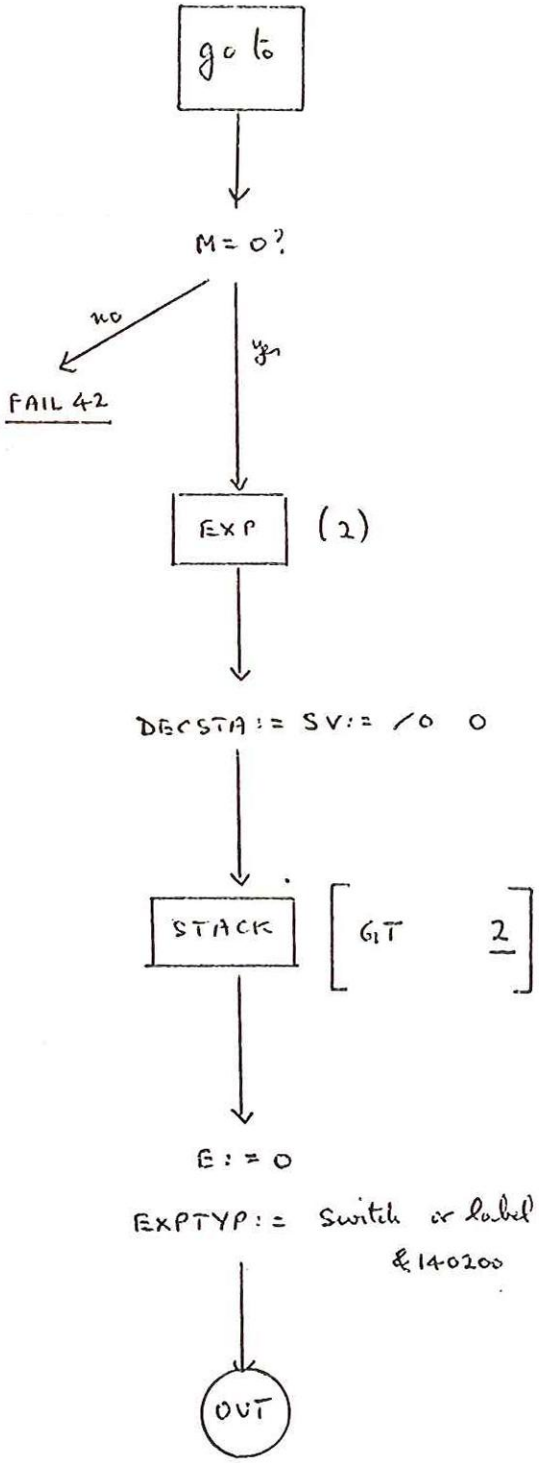

else
TS
?
other
then S
TAKE
(1)
INOUT
(2)
UNSTAK
(2)
ENDSTA
TS
?
other
then E
then S
ALIVO := else E, TYPBCX, 2
FAIL 70
E := 1
ARITH := EXPTYP := 0
ALIVO := else S, 1
RESTO
then, ADDRES
STACK
ALIVO, PP
COMPIL
UJ 8191
update ADDRES
OUT

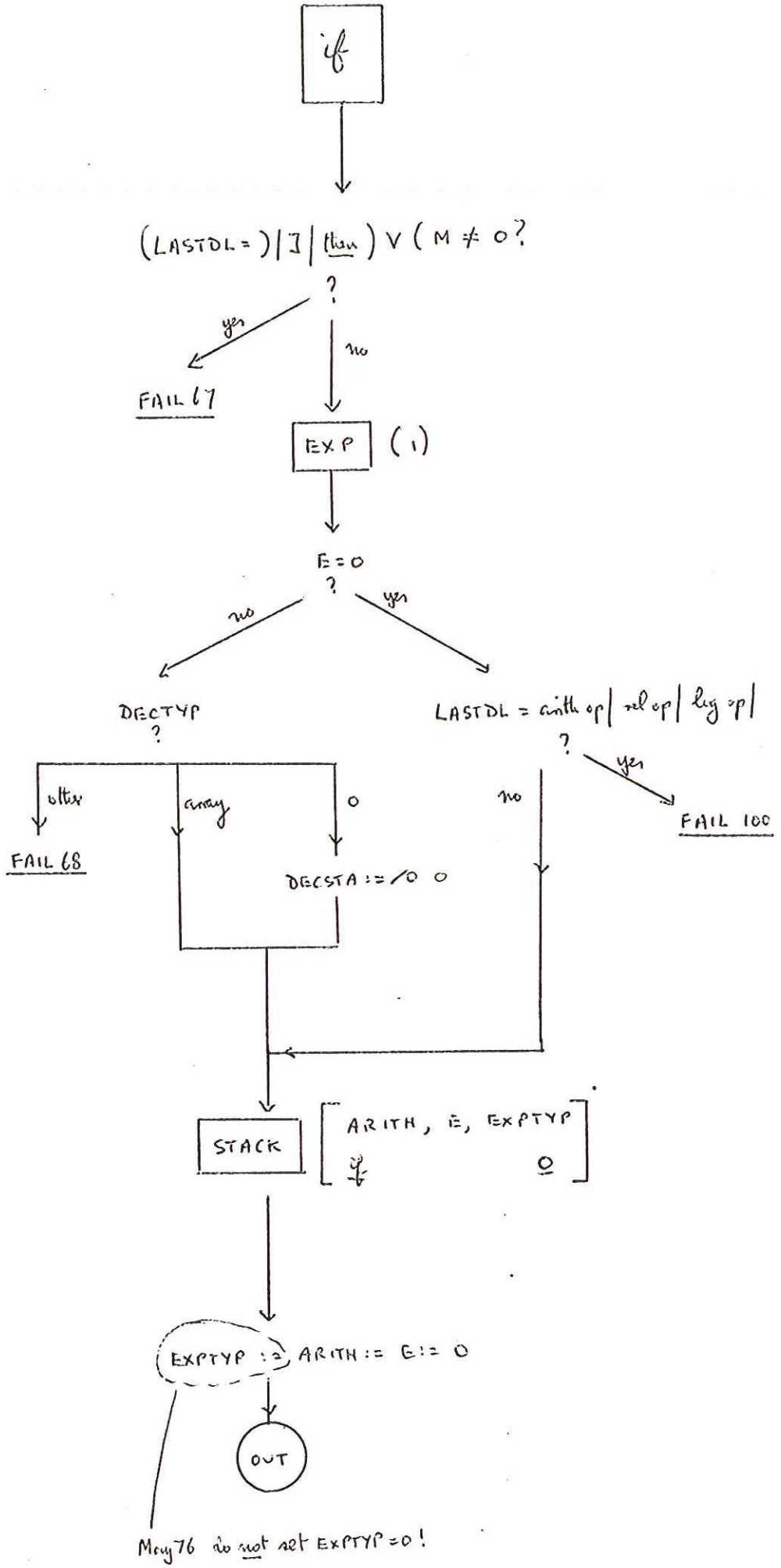
ENT2 is an entry from FAIL

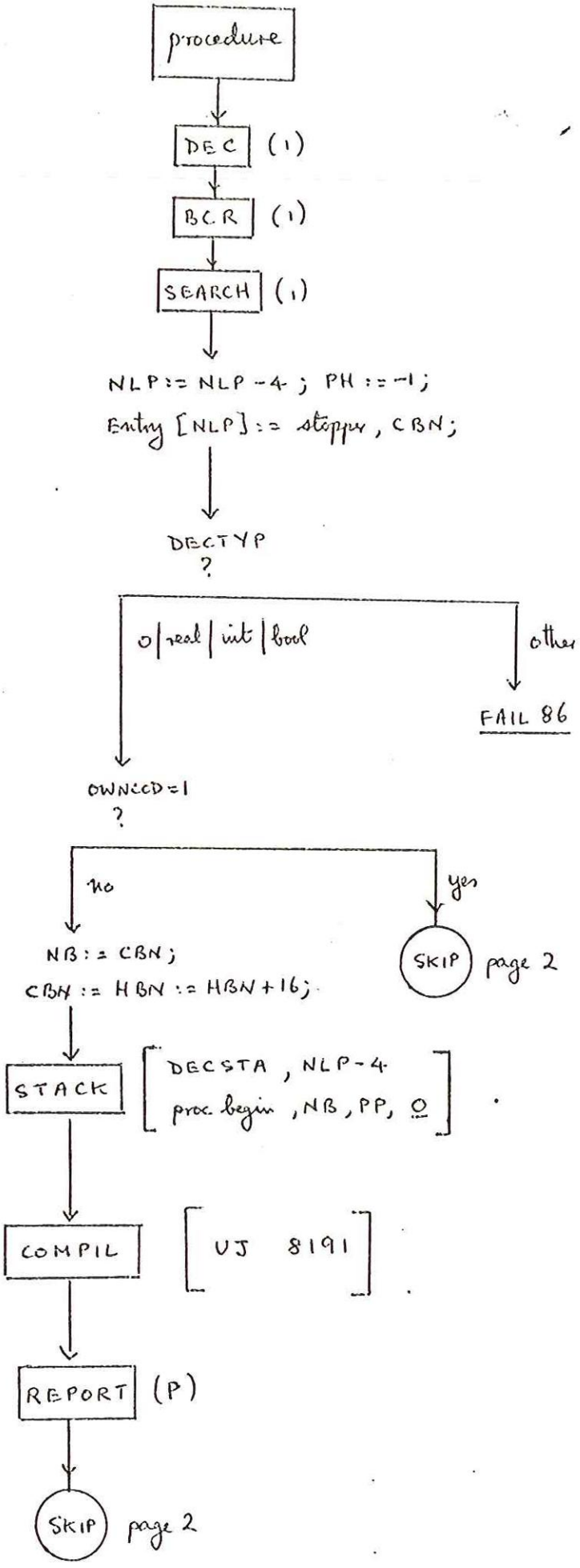



for
(F=0) ∧ (M=0)
?
no
yes
FAIL 44
EXP (2)
DECSTA := /0 0
STACK [for, PP, 0]
NLP := NLP - 4
CHECK
Entry [NLP] := stopper, CBN;
CBN := HBN := HBN + 16;
COMPIL [PRIM FOR / +8191 / CBN / +8191]
F := -1
BCR (1)
DELIM ?
other
:=
FAIL 21
TAKE (0)
STACK [simple, TYPBOX, 0]
E := 0; ARITH := 1
OUT

stopper is -1


go to
M=0?
no
yes
FAIL 42
EXP (2)
DECSTA := SV := /0 0
STACK [GT 2]
E := 0
EXPTYP := Switch or label
&140200
OUT

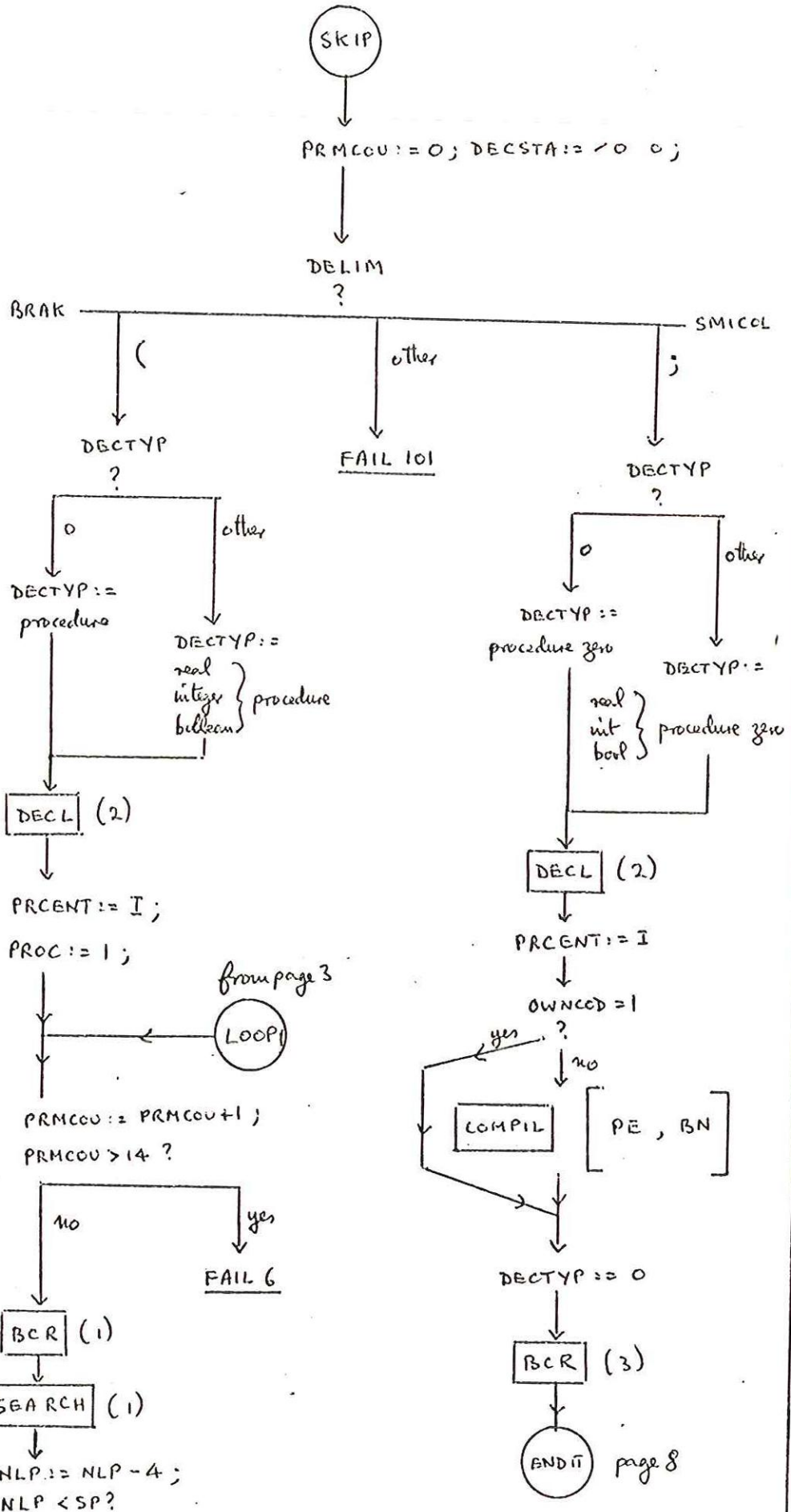

if
(LASTDL =) | I | then) V (M ≠ 0 ?
?
yes
no
FAIL 67
EXP (1)
E = 0
?
no
yes
DECTYP
?
LASTDL = arith op | rel op | log op |
?
yes
FAIL 100
other
array
0
no
FAIL 68
DECSTA := /0 0
STACK
ARITH , E, EXPTYP
1 0
EXPTYP := 2, ARITH := E := 0
OUT
Mary 76 do not set EXPTYP = 0 !



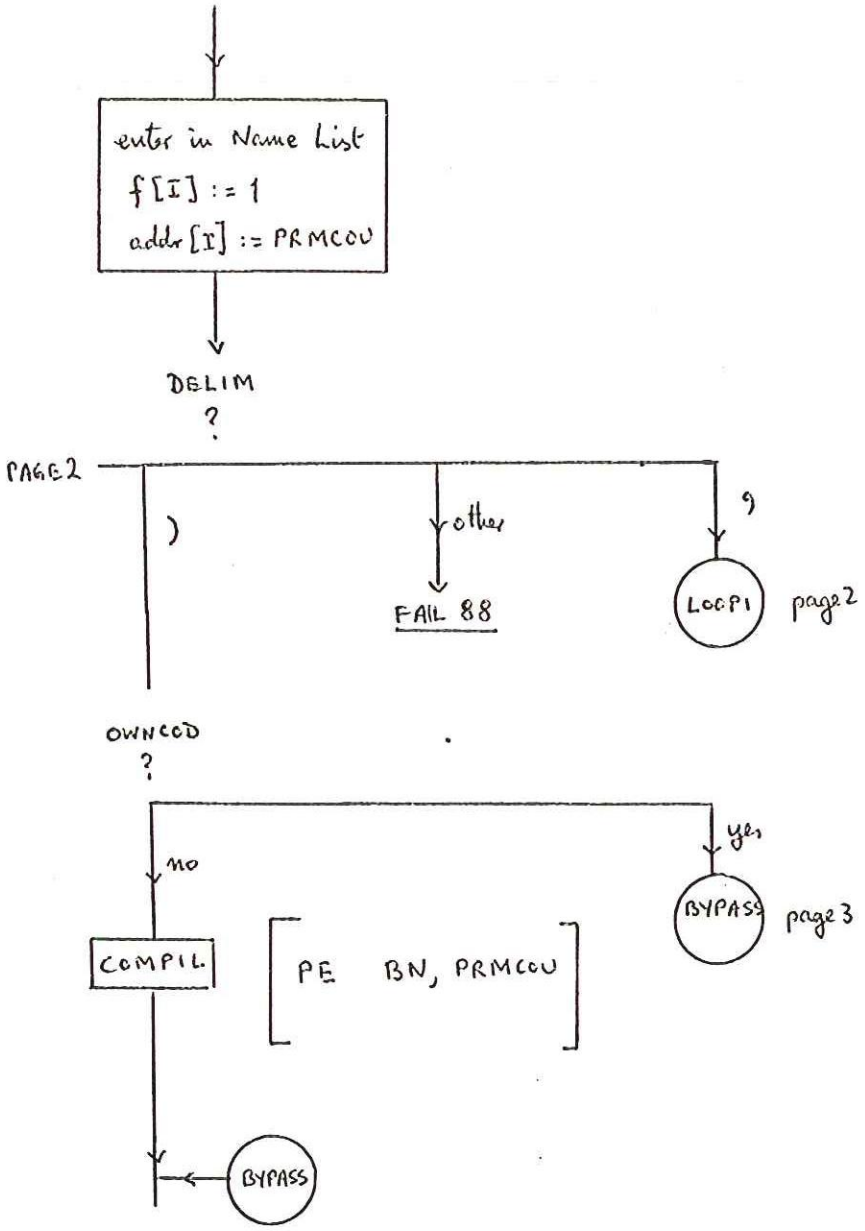
PH is procedure heading marker for FAIL

report procedure name if in report mode

procedure (continued)

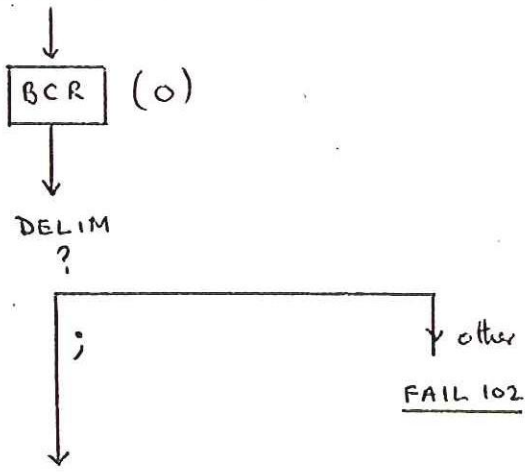


procedure (cont'd)

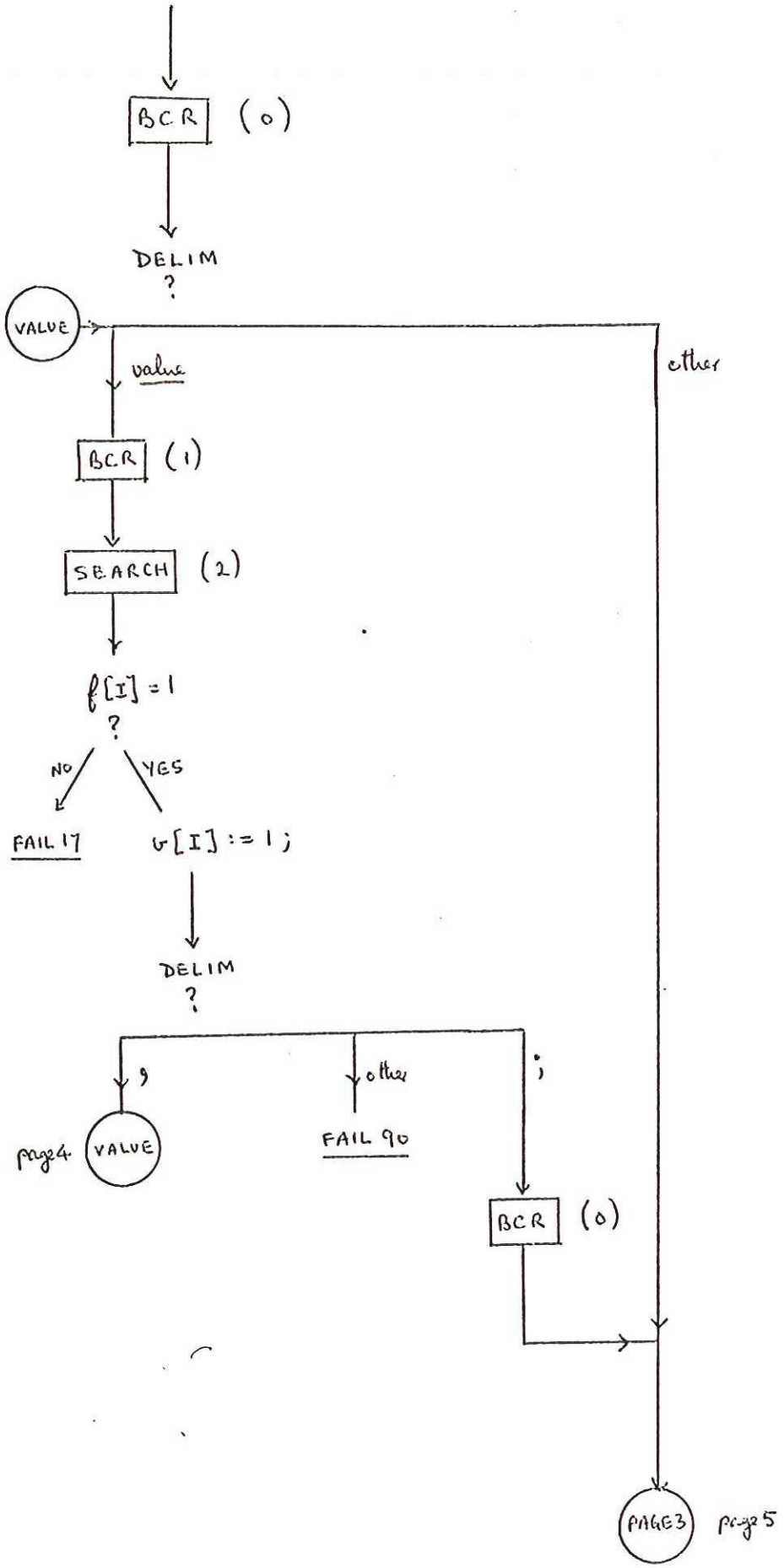


DIM[PRCENT] := PRMCOU;

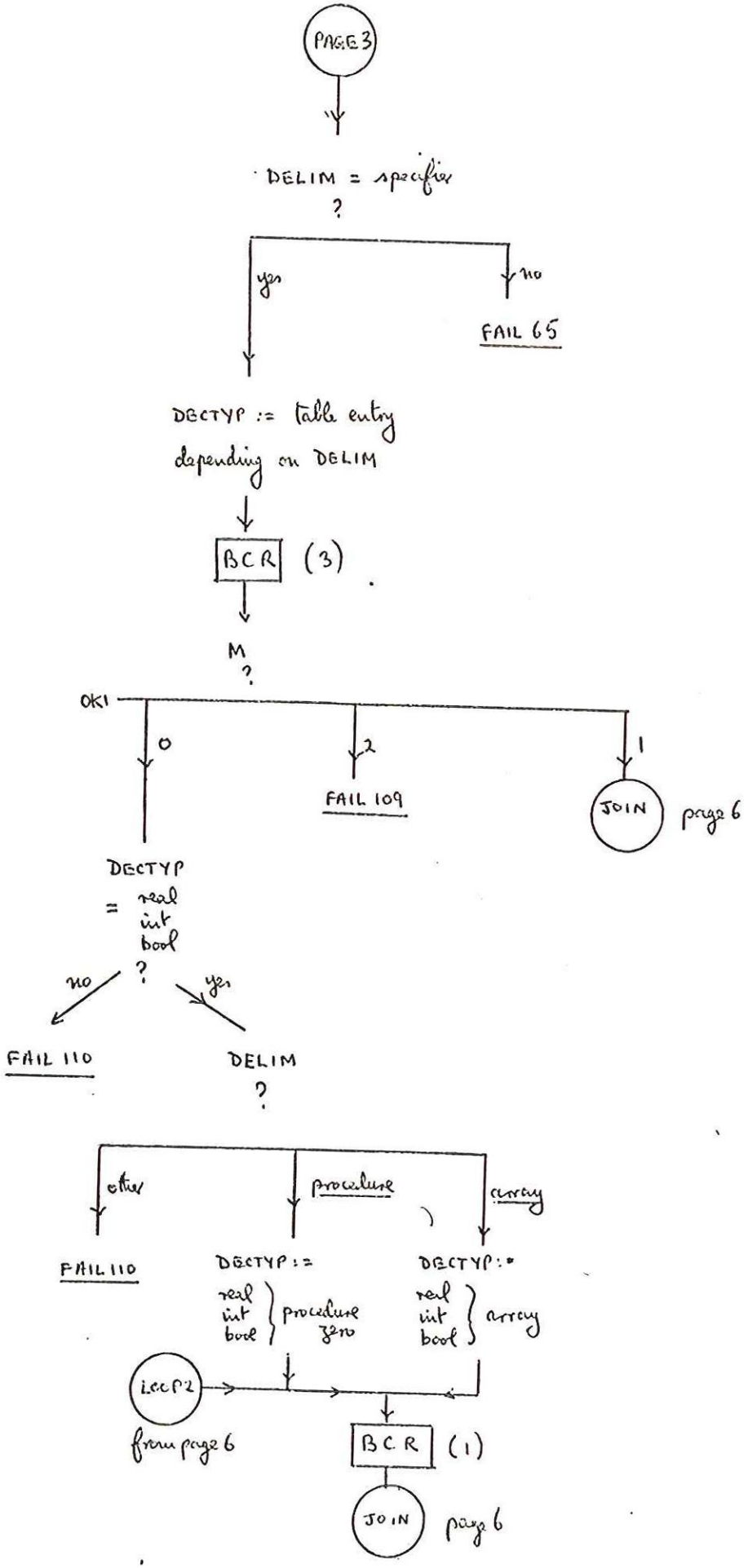
PROC := DECTYP != 0;



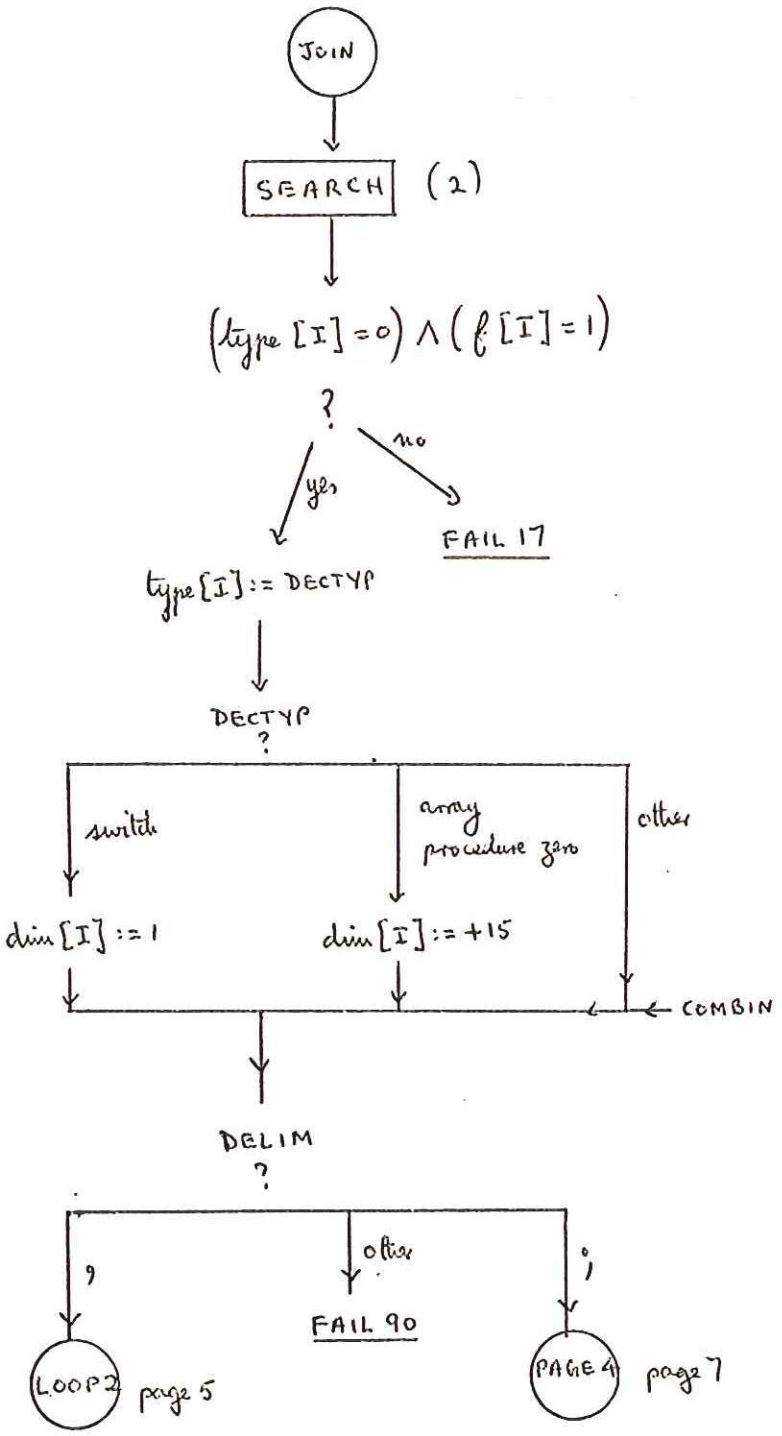
procedure (cont'd)



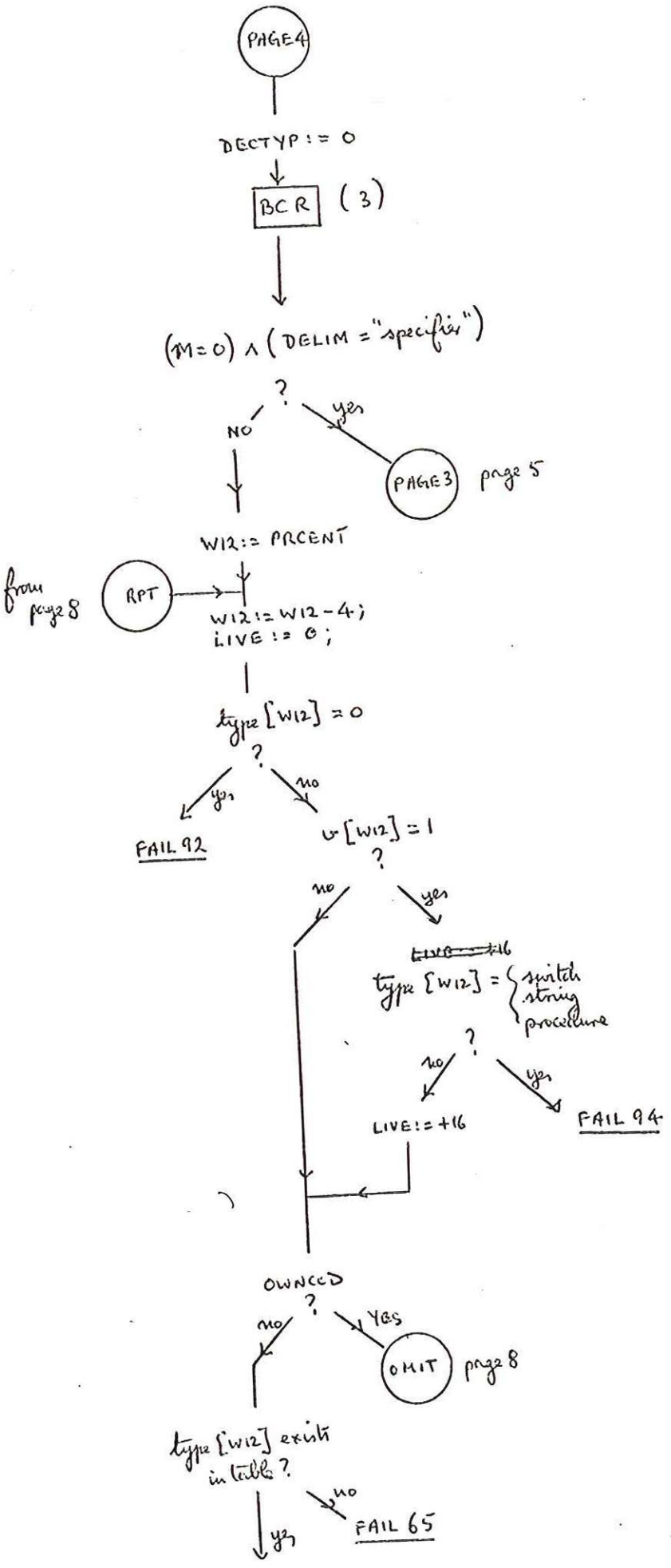
procedure (cont'd)



procedure (cont'd)

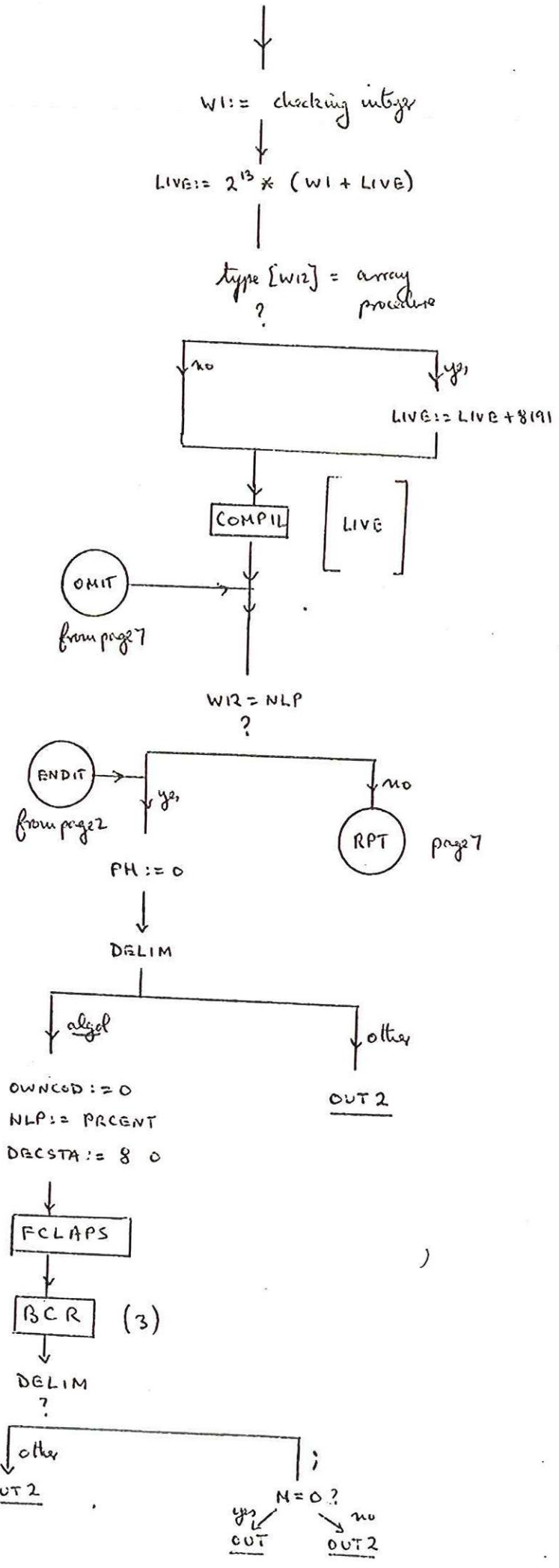


procedure (cont'd)

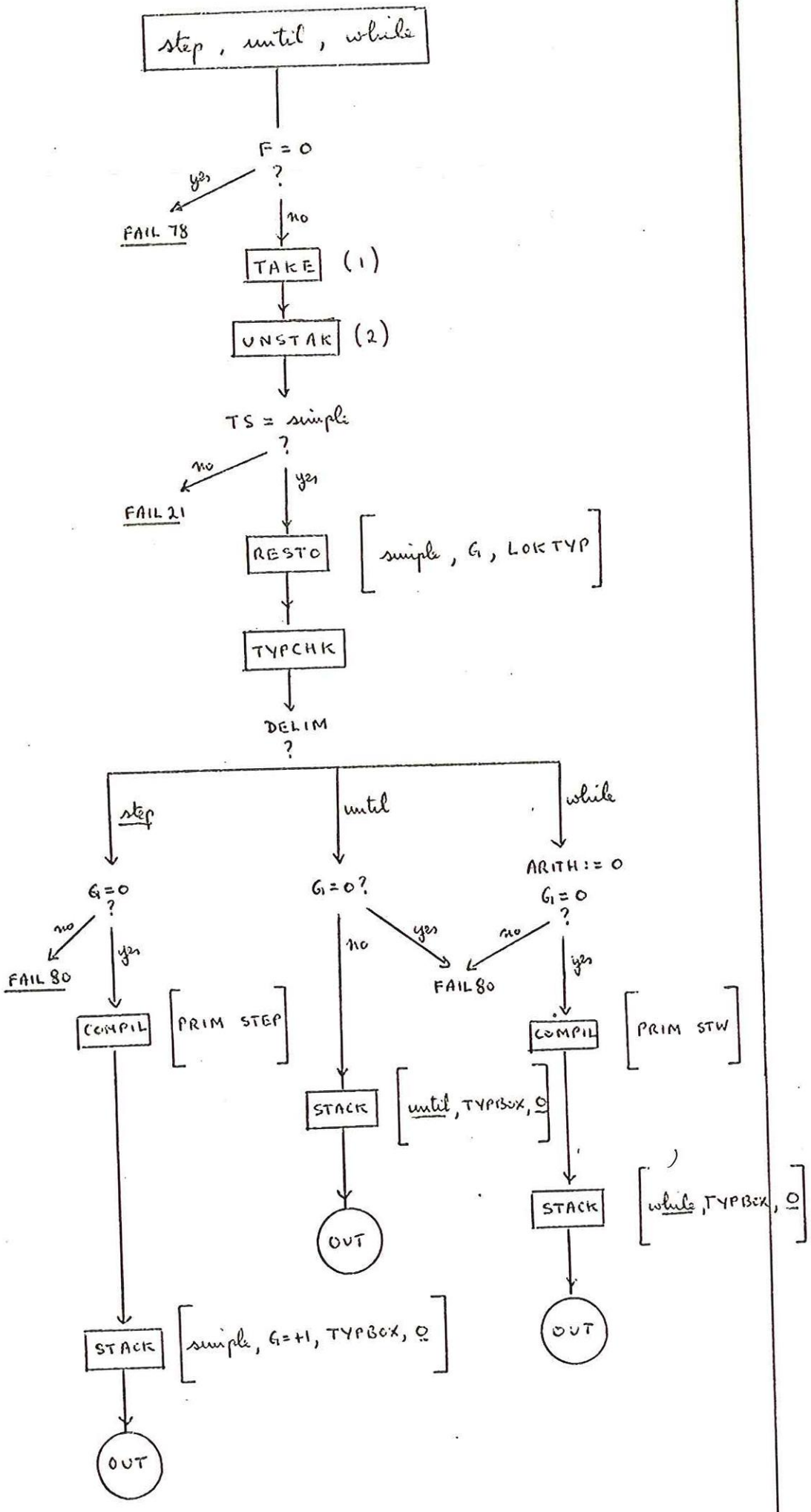


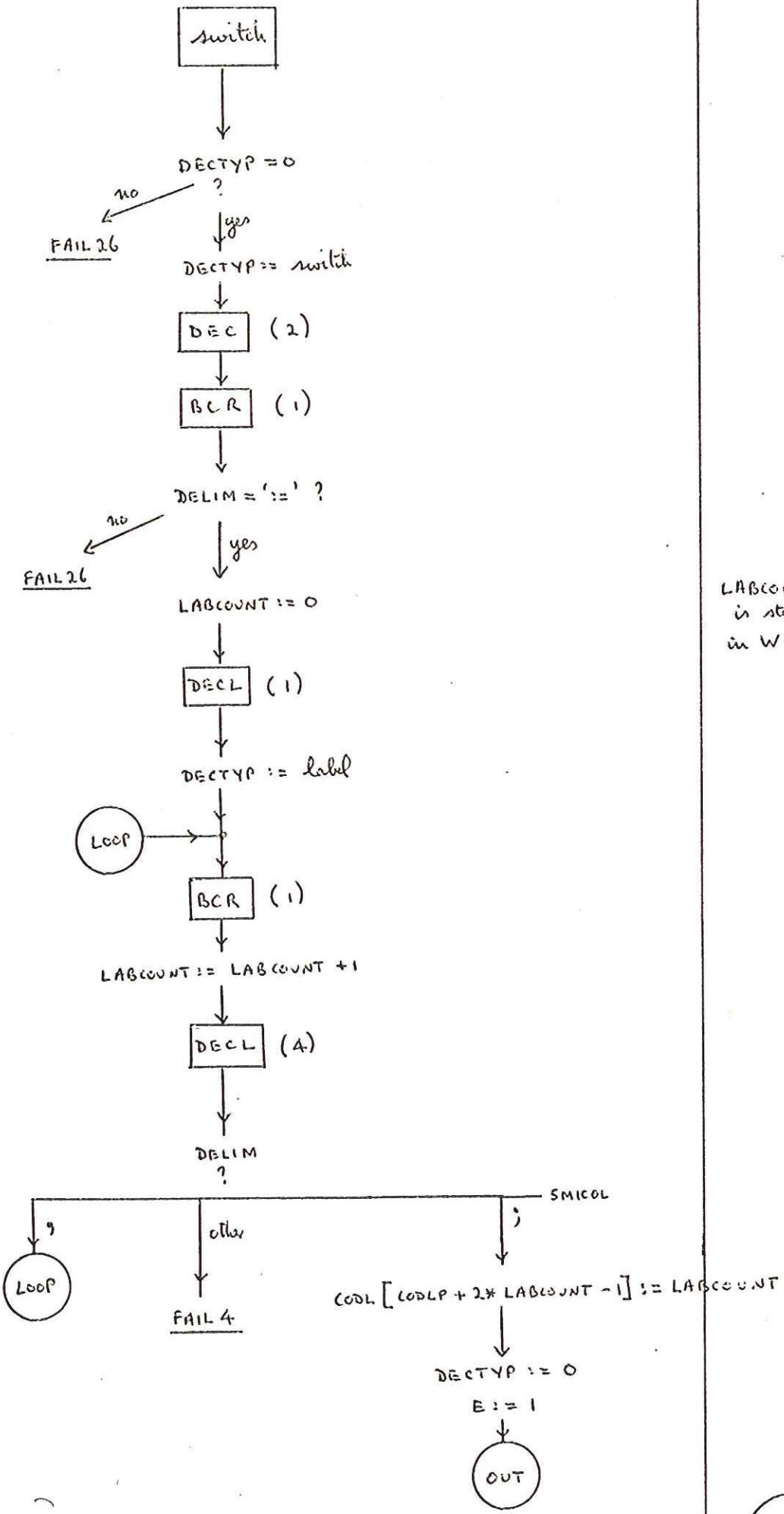
All procedures will be procedure 310 at this stage

procedure (cont'd)

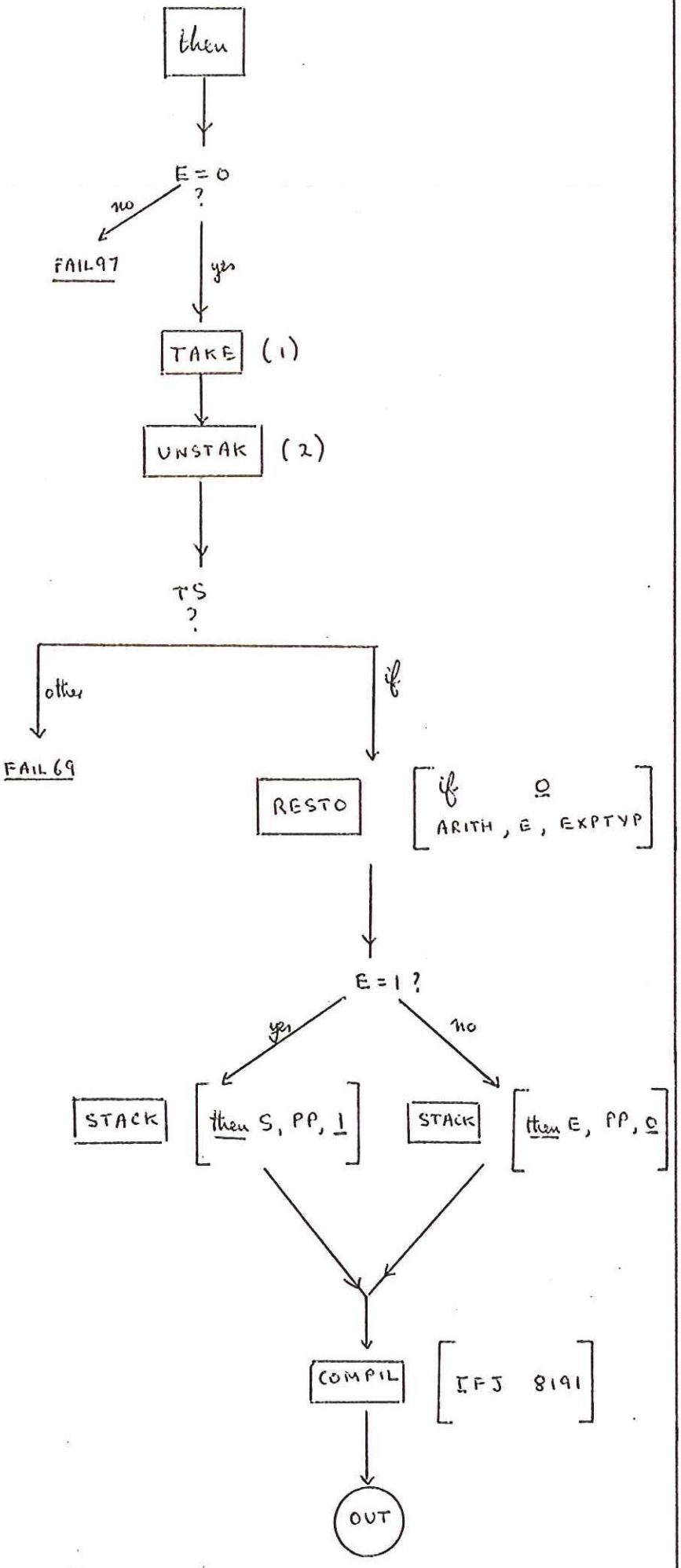


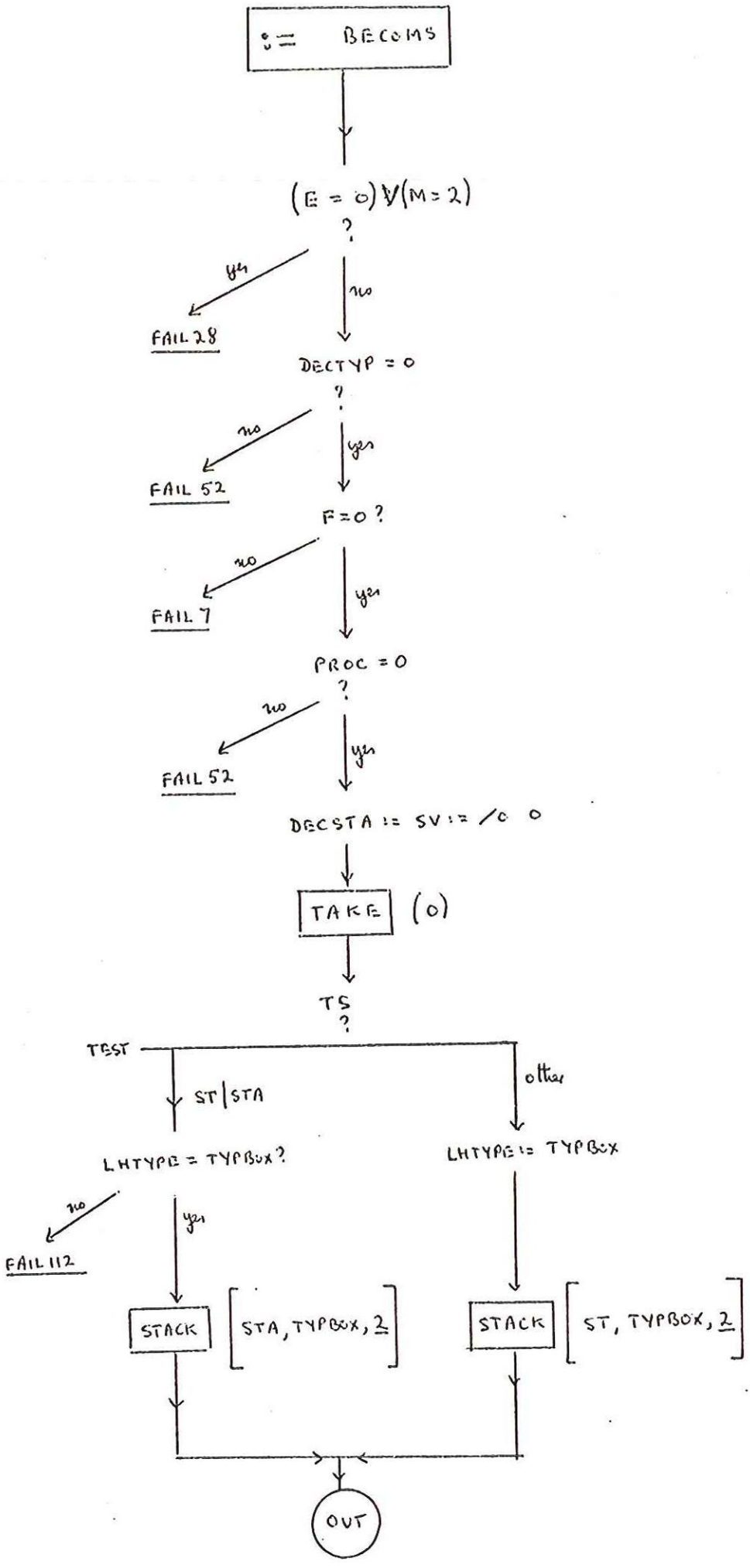
LIVE contains a parameter checking word see pcd manual


step , until , while
F = 0 ?
yes
FAIL 78
no
TAKE (1)
UNSTAK (2)
TS = simple ?
no
FAIL 21
yes
RESTO
[simple , G , LOKTYP]
TYPCHK
DELIM ?
step
until
while
G = 0 ?
no
FAIL 80
yes
COMPIL
[PRIM STEP]
STACK
[simple, G+1, TYPBOX, 0]
OUT
G1 = 0 ?
no
yes
FAIL 80
STACK
[until , TYPBOX, 0]
OUT
ARITH := 0
G1 = 0 ?
no
FAIL 80
yes
COMPIL
[PRIM STW]
STACK
[while , TYPBOX, 0]
OUT

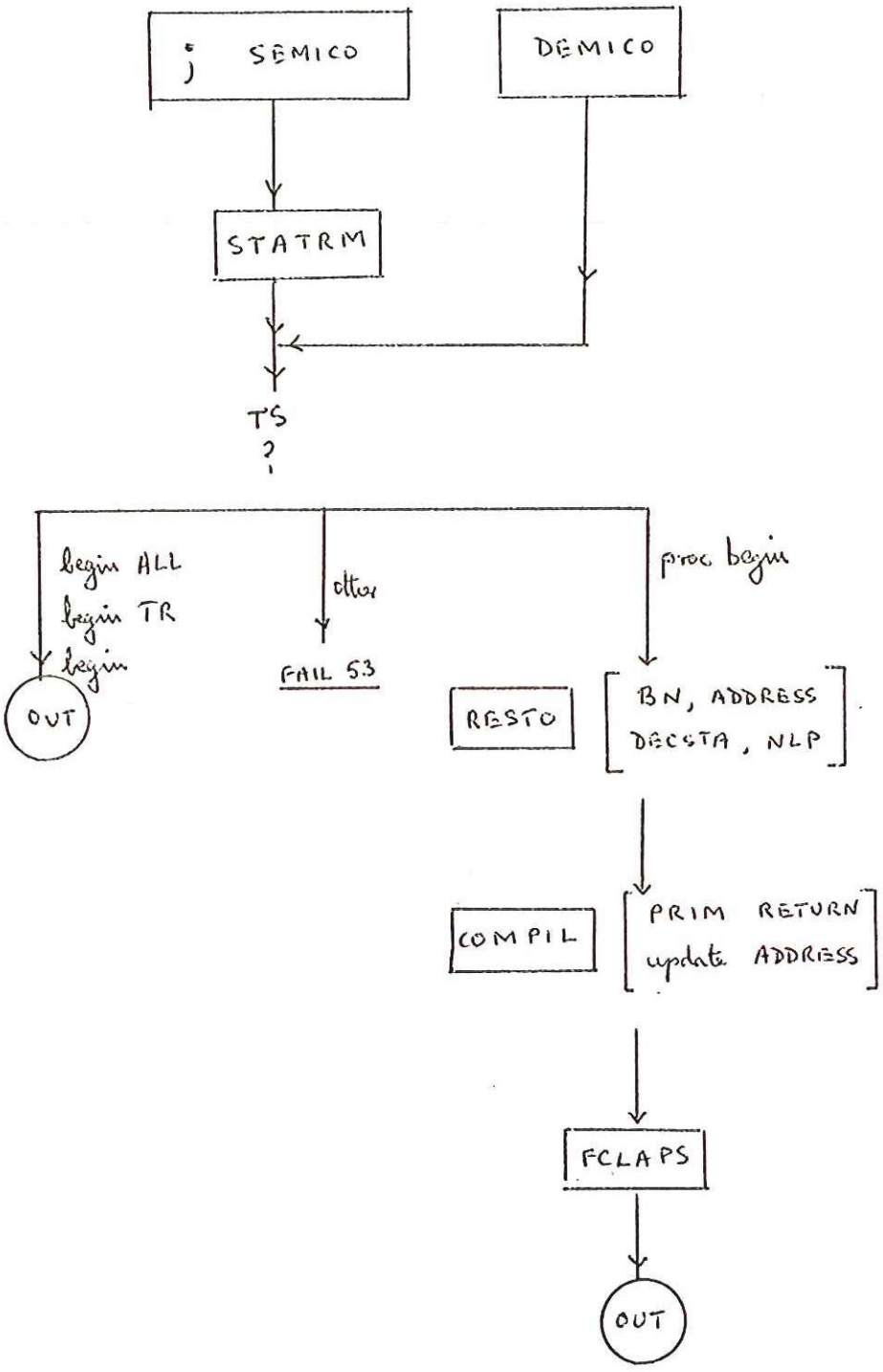


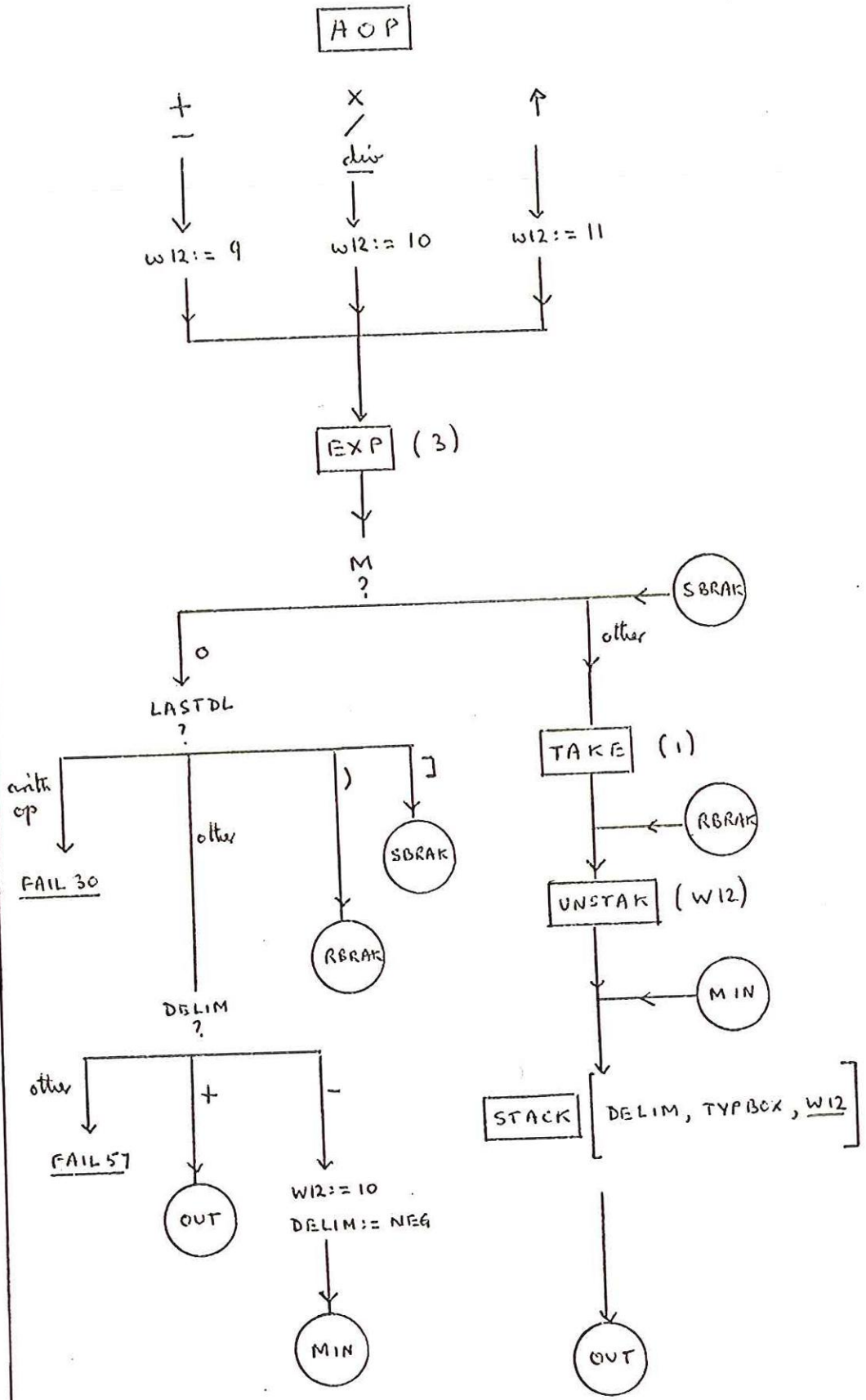
LABCOUNT is stored in W+12


then
E = 0 ?
no
FAIL97
yes
TAKE (1)
UNSTAK (2)
TS ?
other
if
FAIL 69
RESTO
[if 0 ARITH, E, EXPTYP]
E = 1 ?
yes
no
STACK
[then S, PP, 1]
STACK
[then E, PP, 0]
COMPIL
[IFJ 8191]
OUT



SV for subscript variable.


; SEMICO
DEMICO
STATRM
T'S
?
begin ALL
begin TR
begin
other
proc begin
OUT
FAIL 53
RESTO
BN, ADDRESS
DECSTA, NLP
COMPIL
PRIM RETURN
update ADDRESS
FCLAPS
OUT

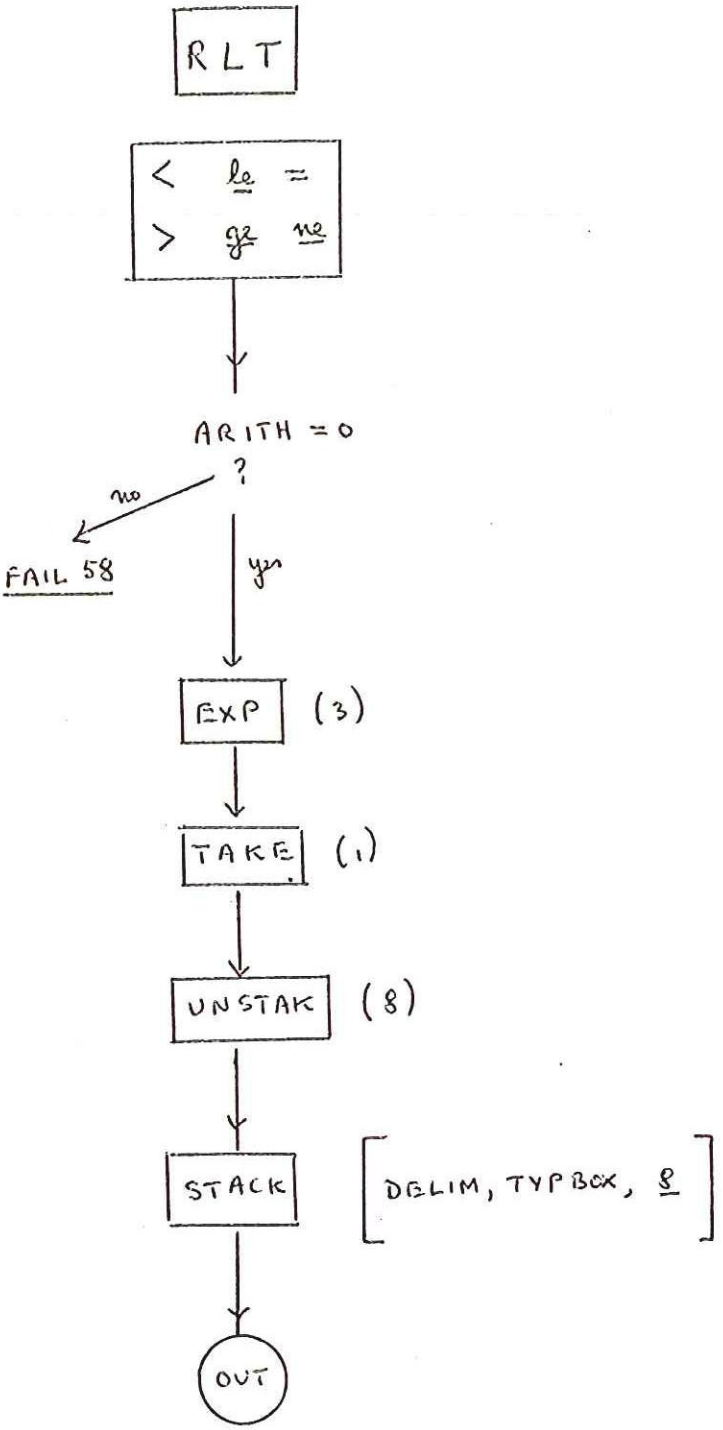


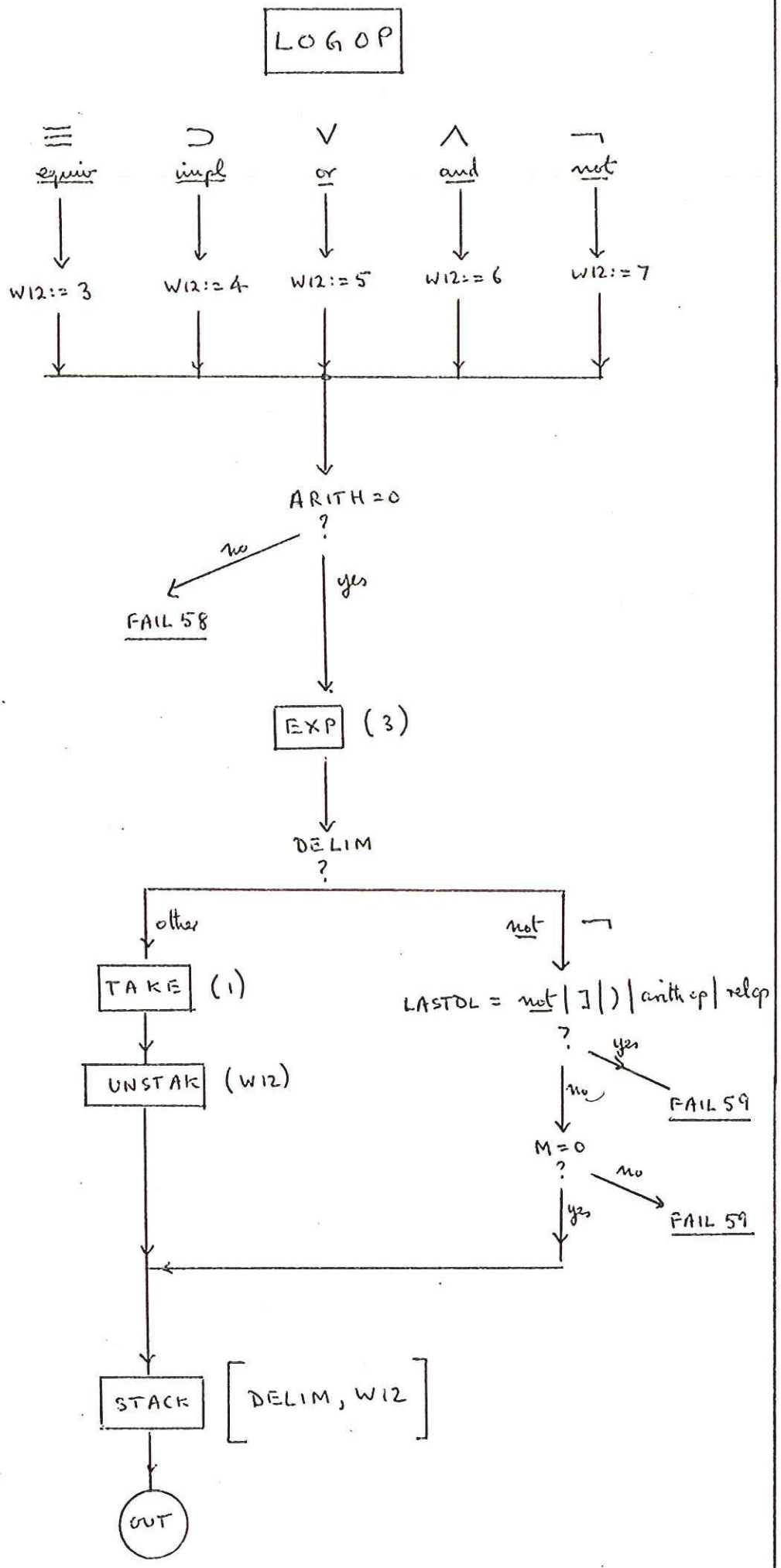
W12 = stack priority

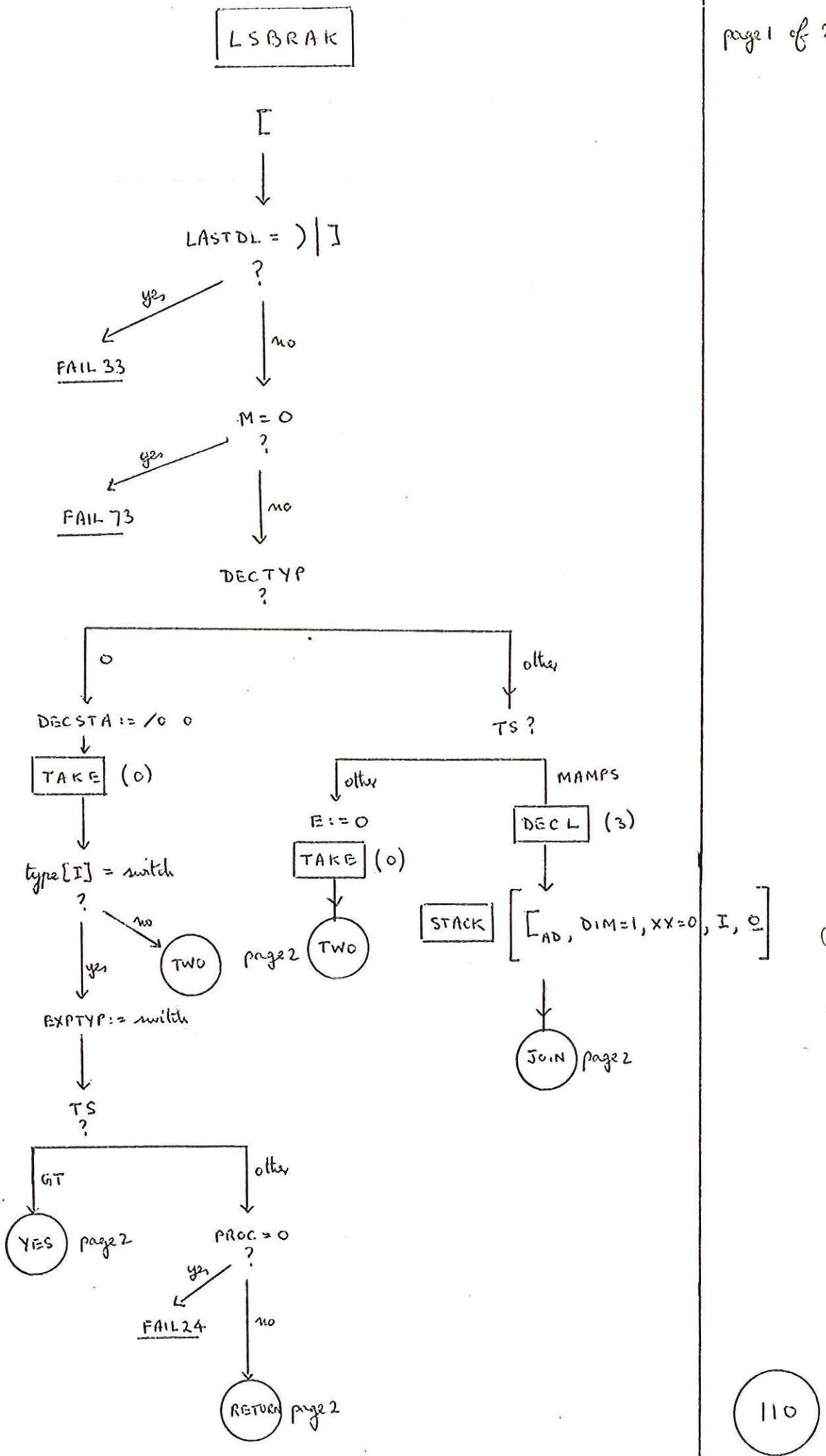
LASTDL+1 has bit pattern to indicate arith, logical or relational operator

unary + is ignored

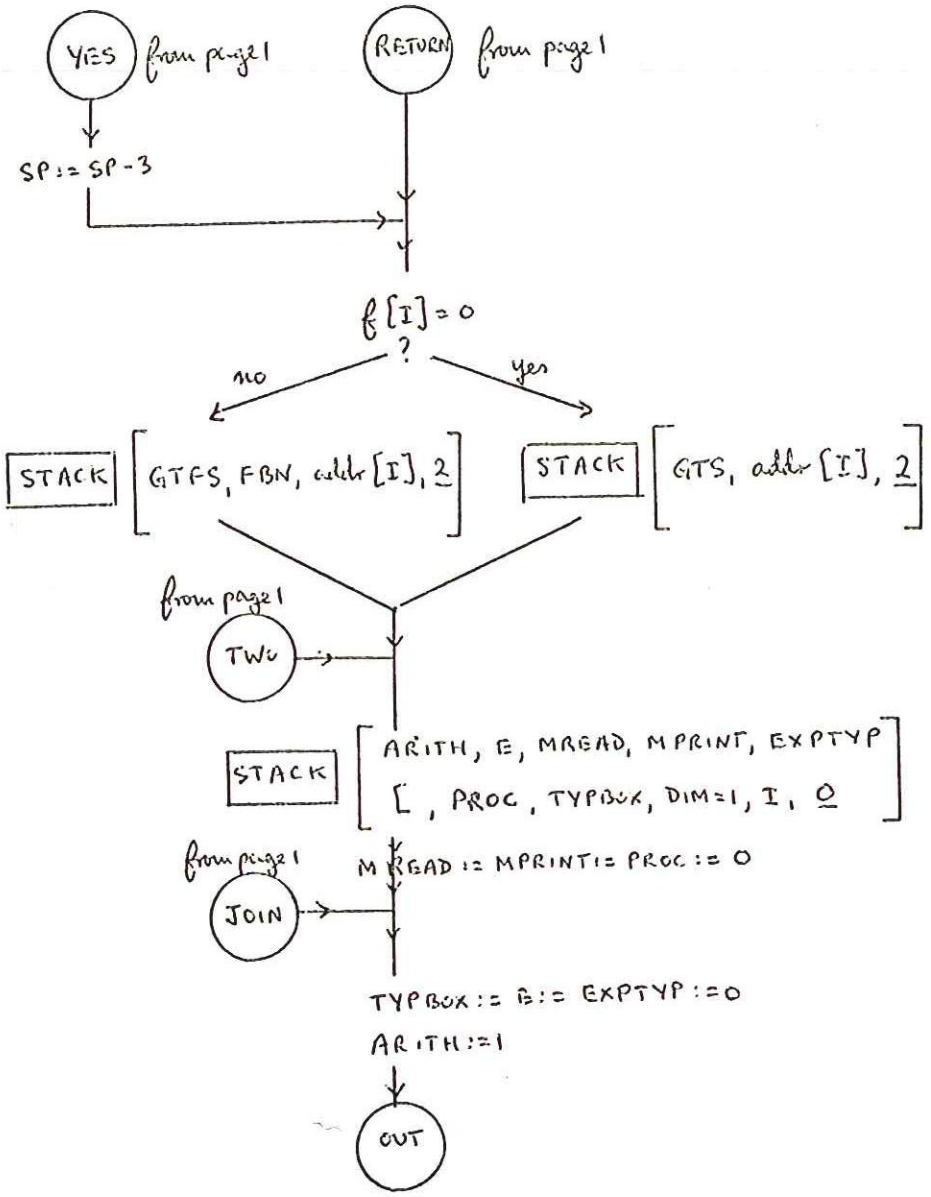
TYPBOX is set by TAKE and UNSTAK

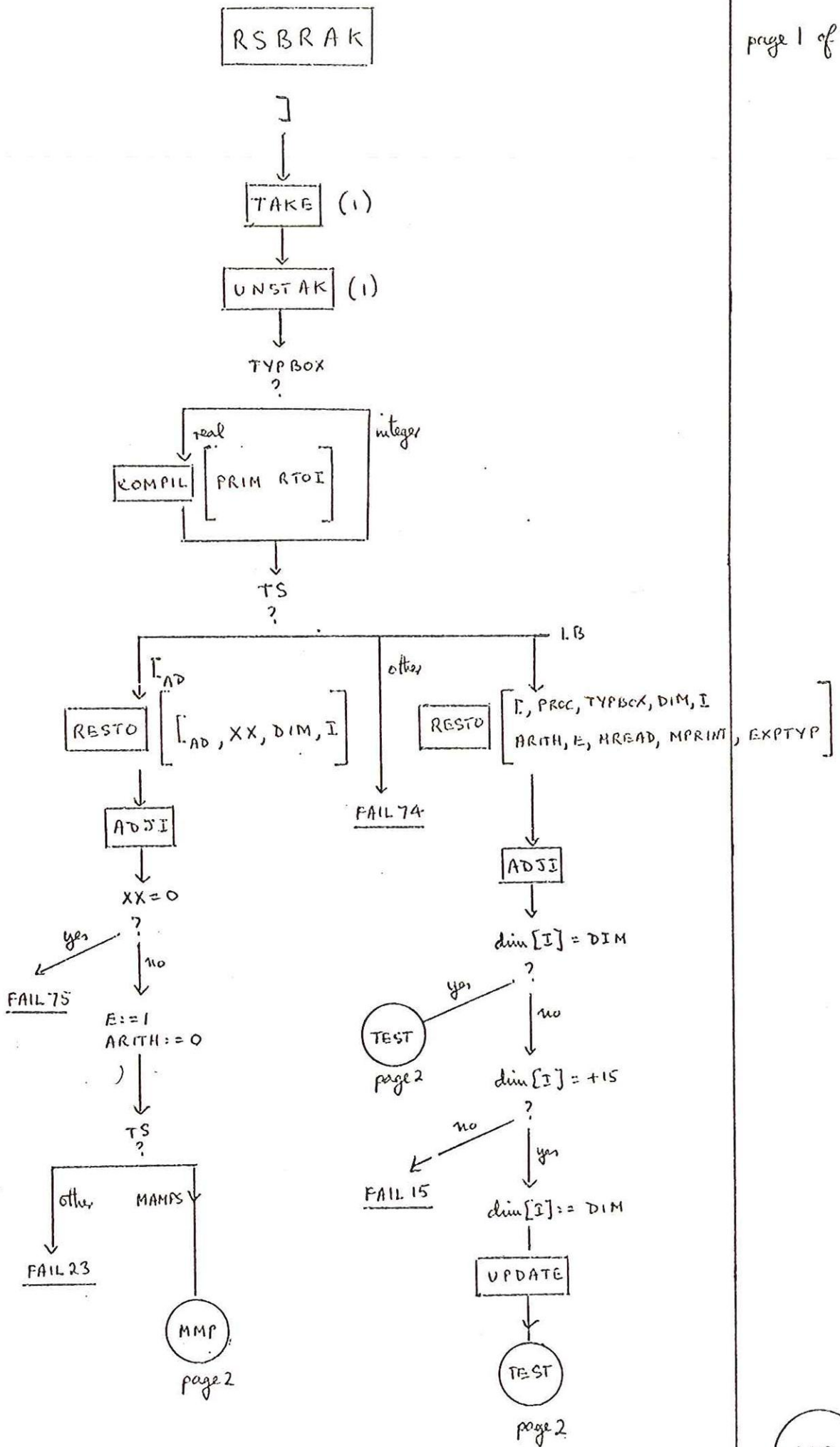

RLT
< le =
> ge ne
ARITH = 0
?
no
FAIL 58
yes
EXP (3)
TAKE (1)
UNSTAK (8)
STACK
[DELIM, TYPBOX, 8]
OUT


LOGOP
≡ equiv
⊃ impl
∨ or
∧ and
¬ not
W12:=3
W12:=4
W12:=5
W12:=6
W12:=7
ARITH=0 ?
no
yes
FAIL 58
EXP (3)
DELIM ?
other
not ¬
TAKE (1)
UNSTAK (W12)
LASTOL = not |] |) | arith op | relop ?
yes
FAIL 59
no
M=0 ?
no
FAIL 59
yes
STACK [DELIM, W12]
OUT

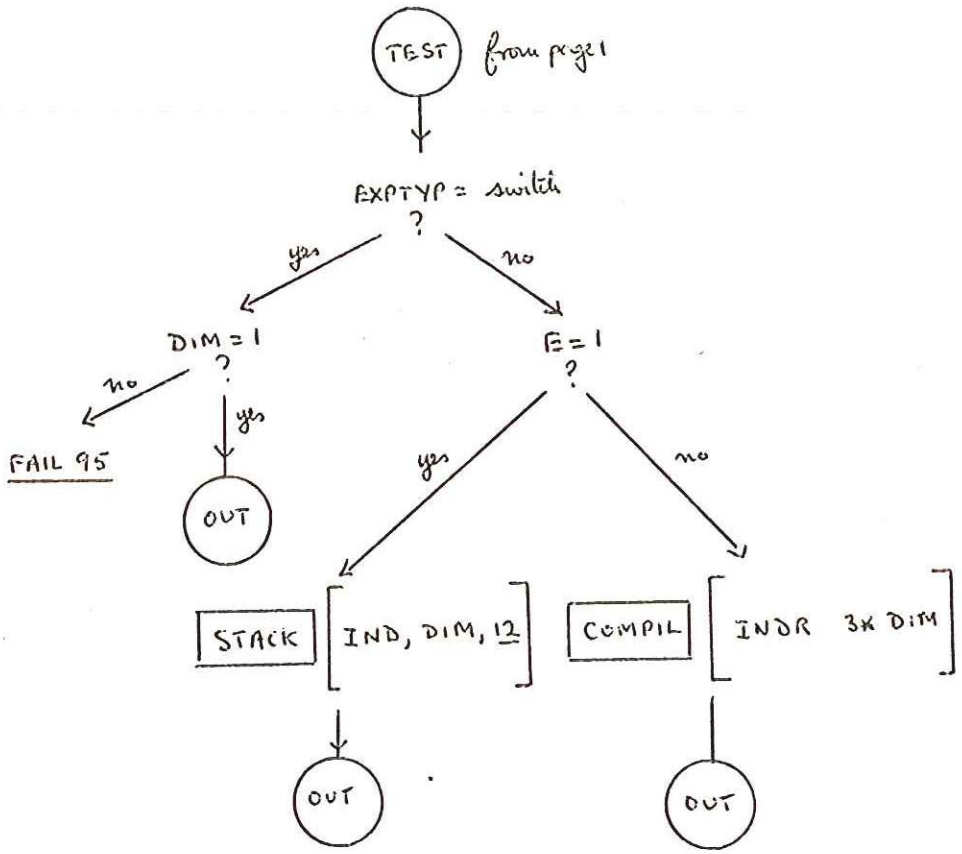

LSBRAK
[
LASTDL =) |]
?
yes
no
FAIL 33
M = 0
?
yes
no
FAIL 73
DECTYP
?
0
other
DECSTA := /0 0
TAKE (0)
type[I] = switch
?
no
yes
TWO
EXPTYP := switch
TS
?
GT
other
YES page 2
PROC = 0
?
yes
no
FAIL 24
RETURN page 2
TS ?
other
MAMPS
E := 0
TAKE (0)
page 2 TWO
DECL (3)
STACK
[[AD, DIM=1, XX=0, I, 0]
JOIN page 2

LSBRAK contd

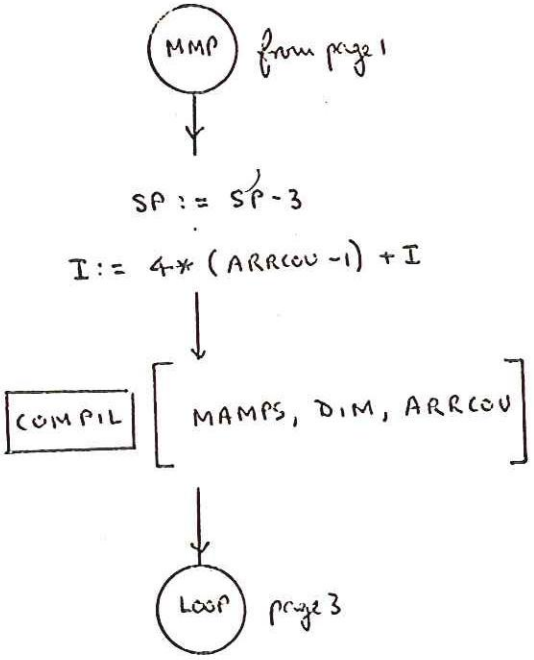



RSBRAK
]
TAKE (1)
UNSTAK (1)
TYPBOX ?
real
integer
COMPIL
PRIM RTOI
TS ?
I.B
[.AD
other
RESTO
[.AD, XX, DIM, I
RESTO
[., PROC, TYPBOX, DIM, I
ARITH, E, MREAD, MPRINT, EXPTYP
FAIL 74
ADJI
XX=0 ?
yes
no
FAIL 75
E:=1
ARITH:=0
)
TS ?
other
MAMPS
FAIL 23
MMP
page 2
ADJI
dim[I] = DIM ?
yes
no
TEST
page 2
dim[I] = +15 ?
no
yes
FAIL 15
dim[I]:= DIM
UPDATE
TEST
page 2

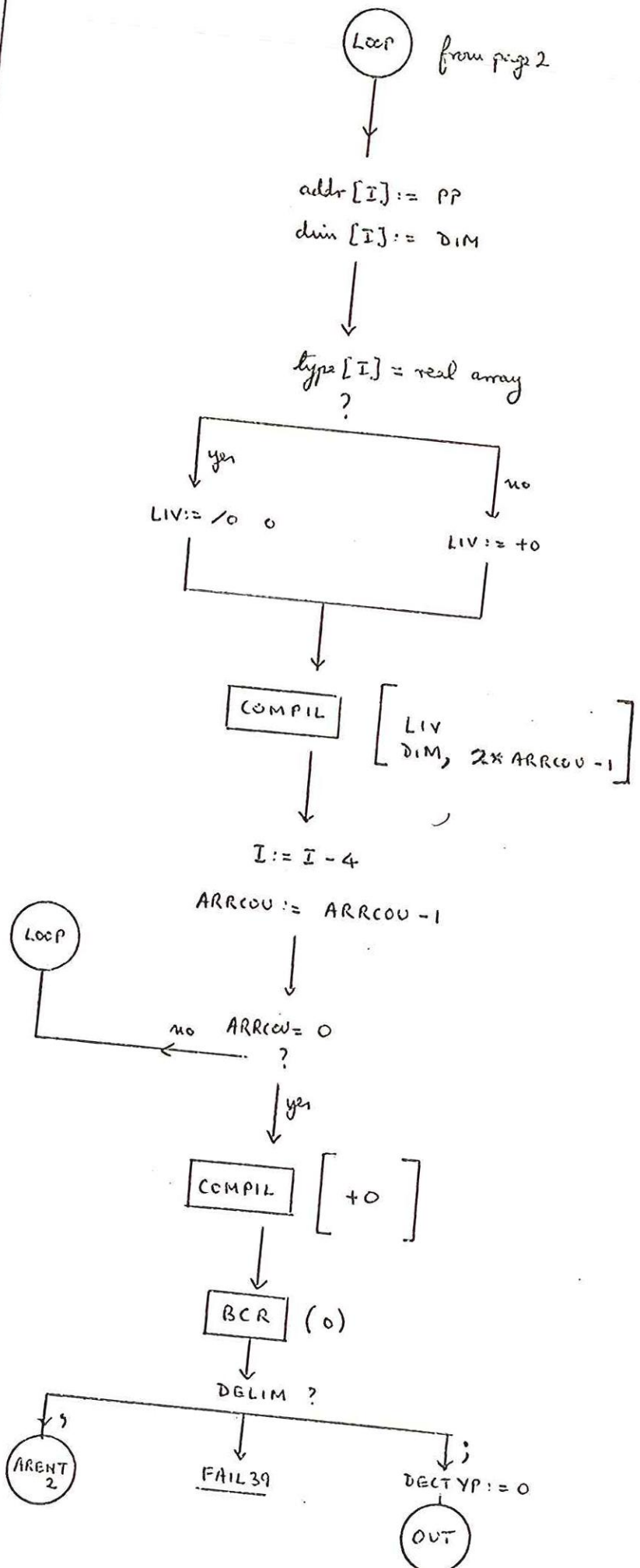
RSBRAK cont'd



DIM = dim [I]



RSBRAK contd



enter details in namelist

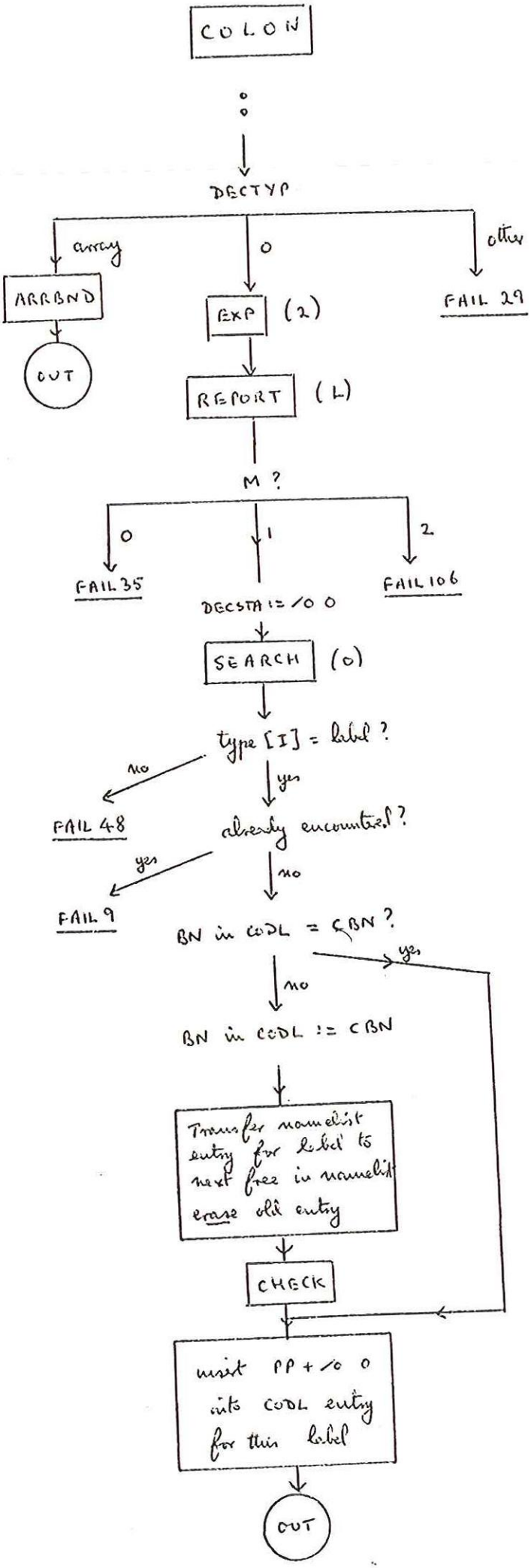
compile array pair

I is reset to next array name

compile shared map pointer (fills in at run time)

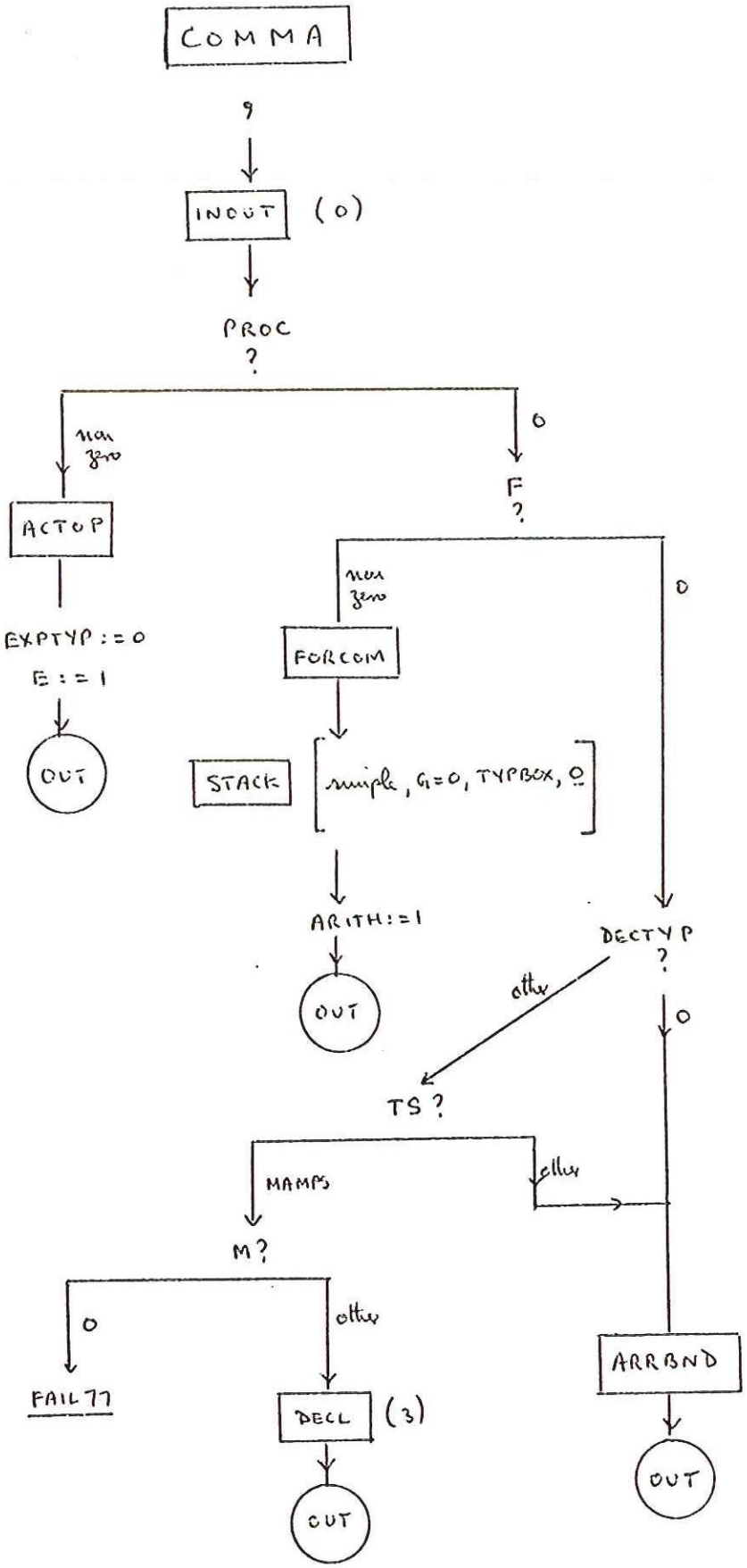
114

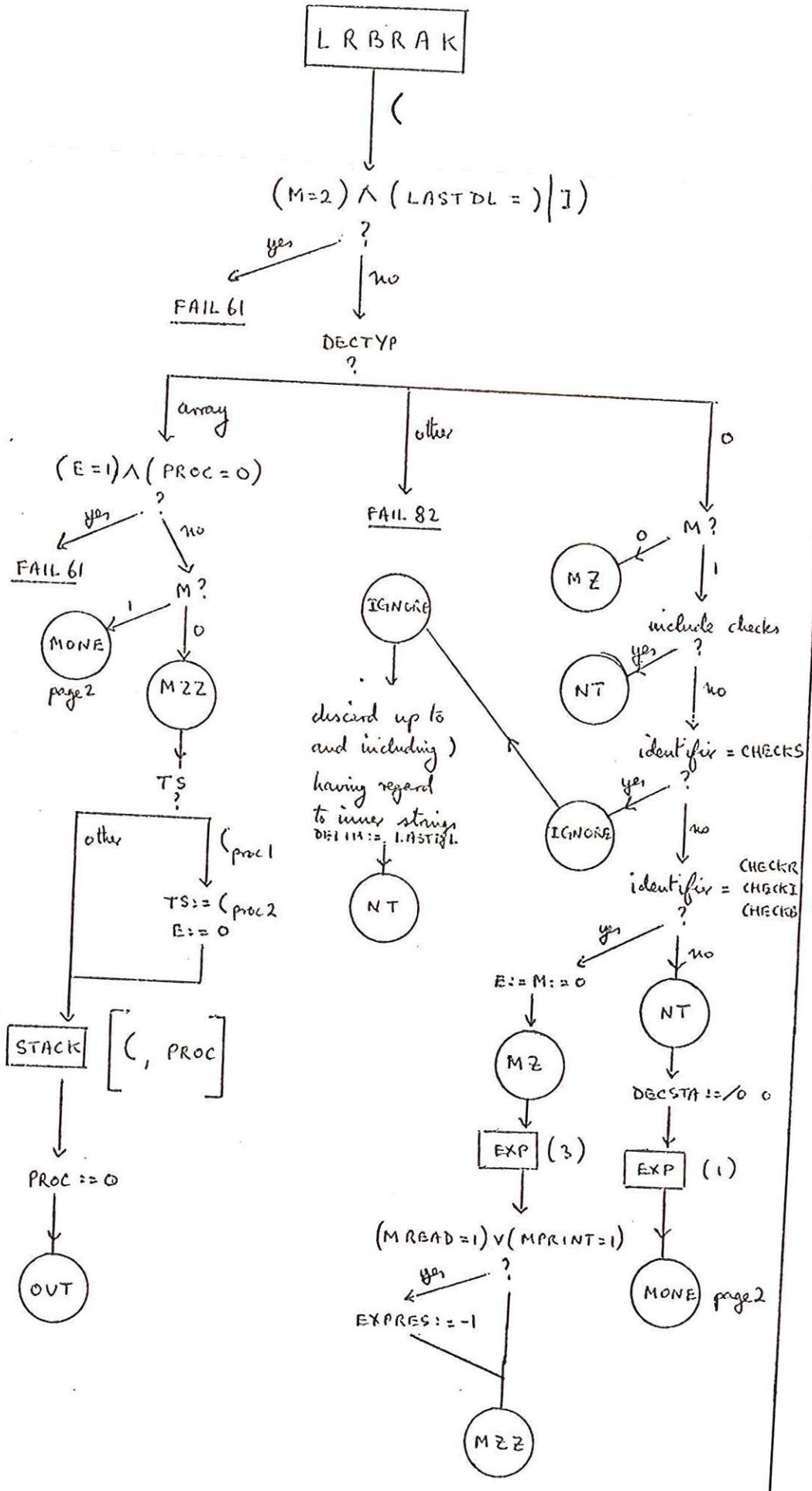
ARENT 2 is an entry to array



force block name of current block. check at run time.

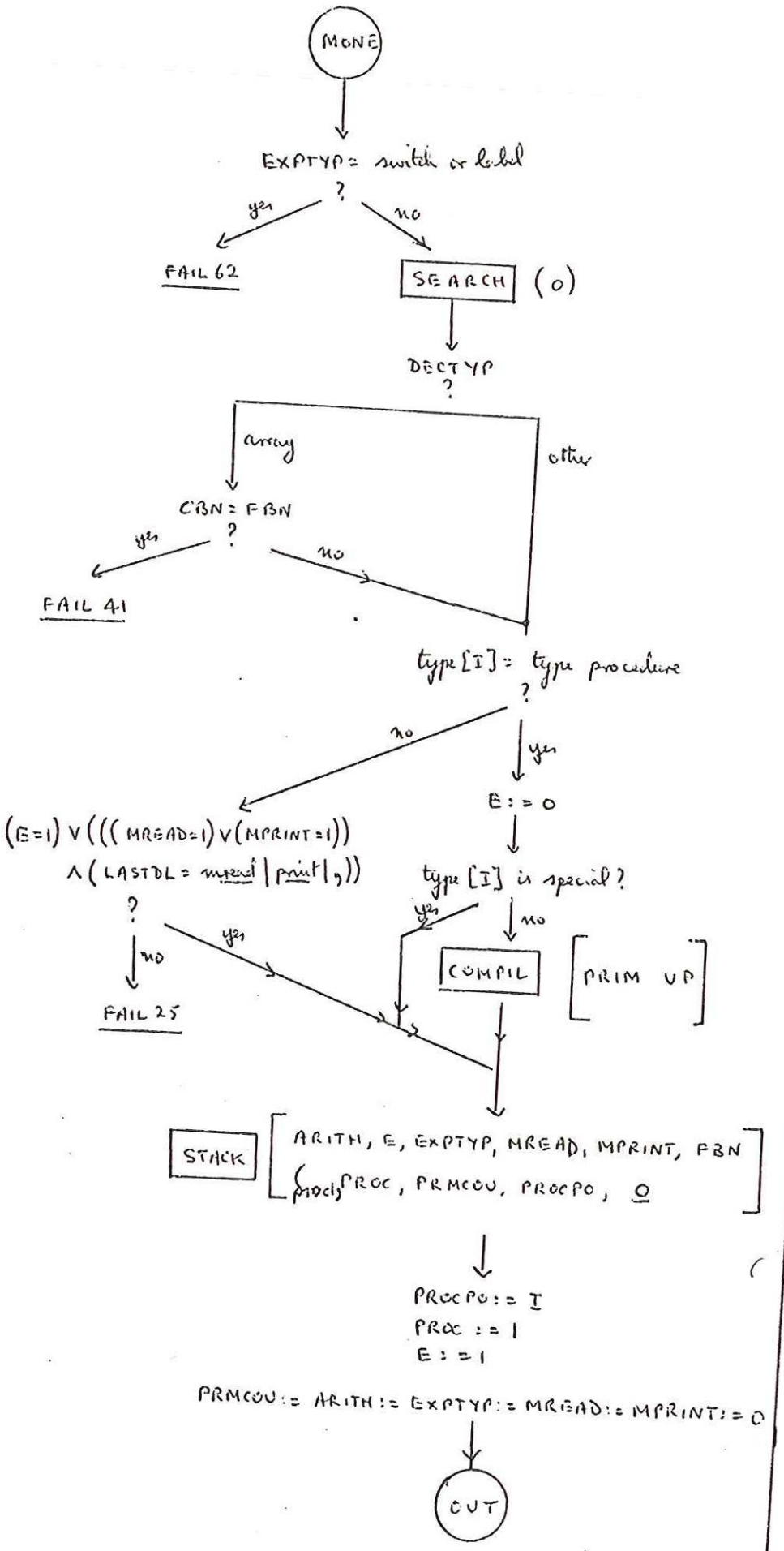
Check for overlap with CODL


COMMA
9
INOUT
(0)
PROC
?
non zero
0
ACTOP
F
?
EXPTYP := 0
E := 1
OUT
non zero
0
FORCOM
STACK
simple, G=0, TYPBOX, 0
ARITH := 1
OUT
DECTYP
?
other
0
TS ?
MAMPS
other
M?
0
other
FAIL 77
DECL
(3)
OUT
ARRBND
OUT



checks included if OPTION 4 bit present

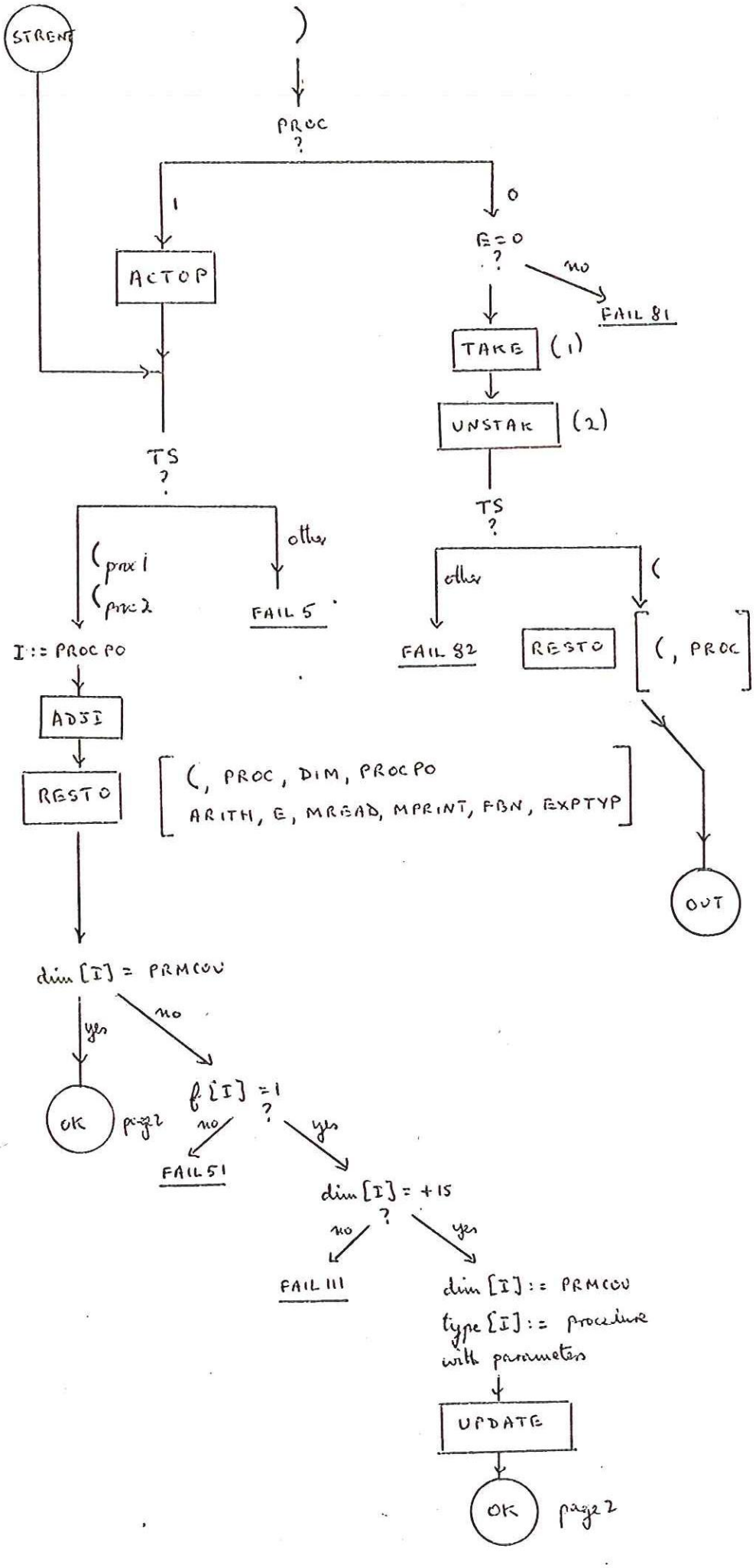
LRBRAK contd



special means compilable as a PRIM and suppress PRIM UP

(Proc1 is /2 4094

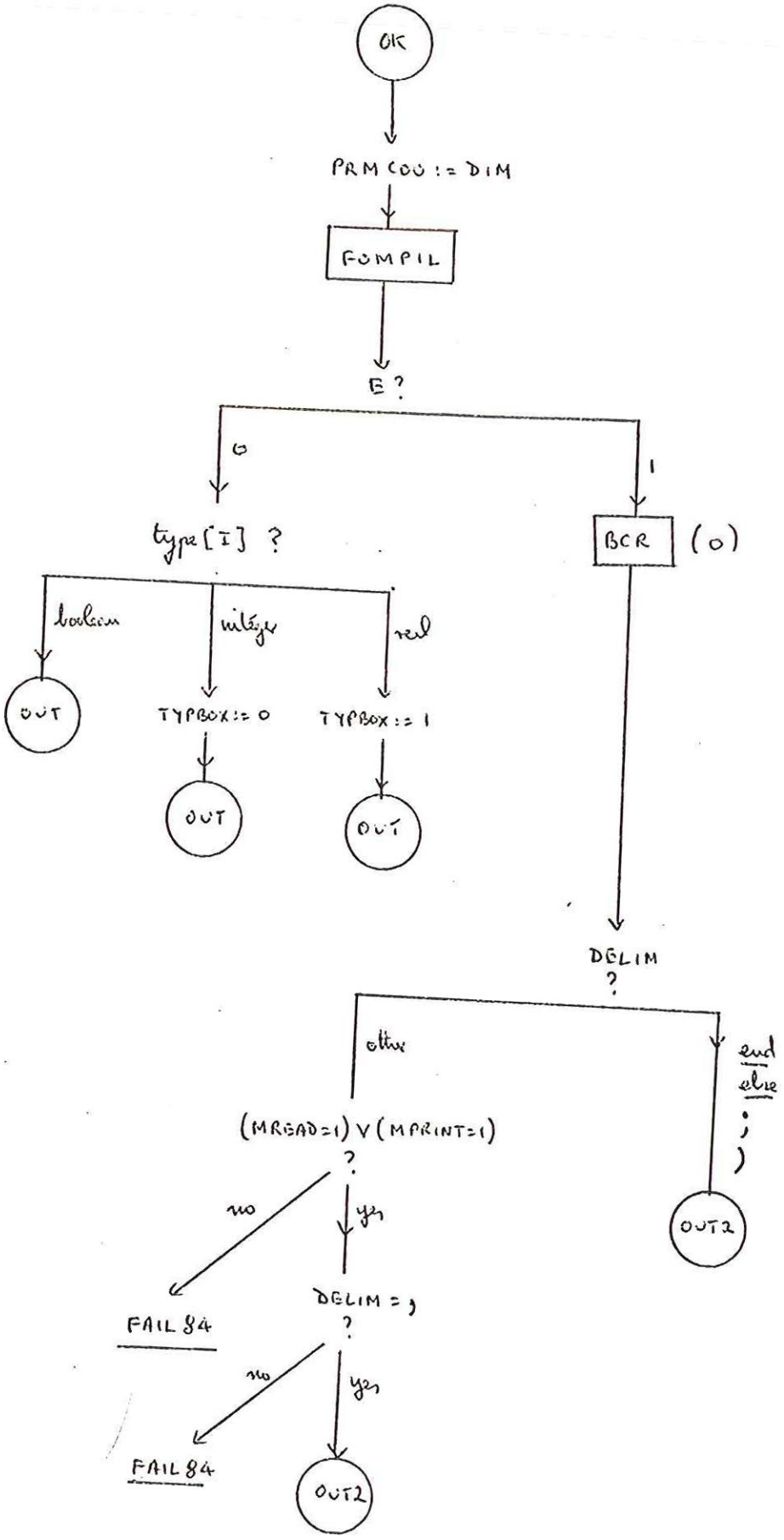
RRBRAK

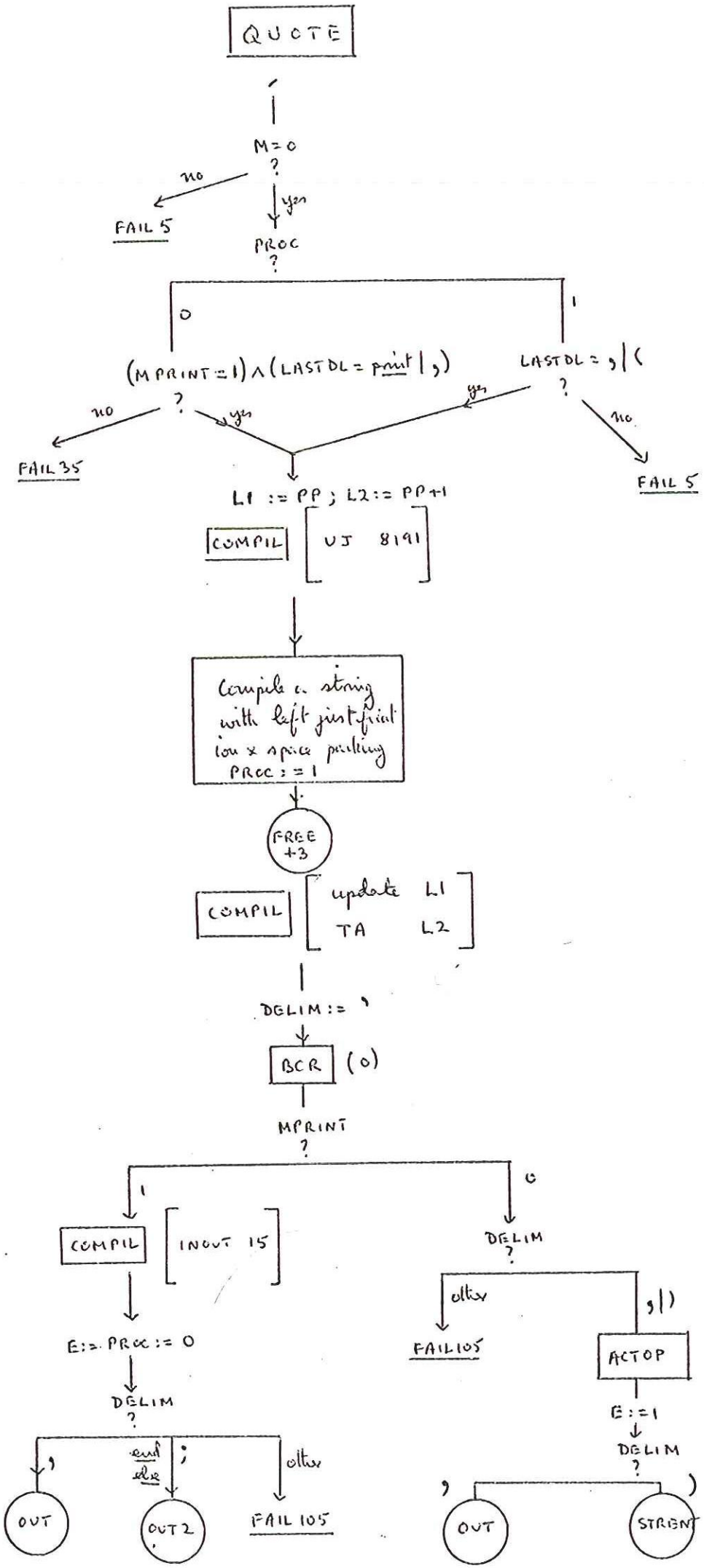


STRENT is an entry from opening string quote

This (may be (proc 1 or (proc 2

RRBRAK contd

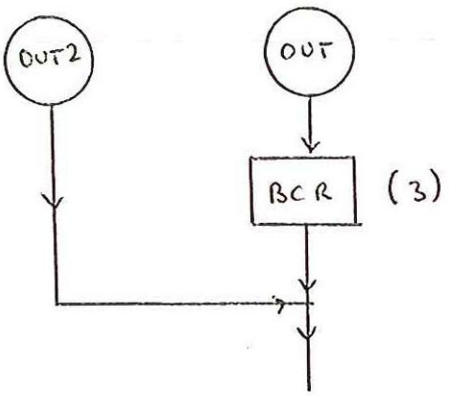




STRENT is entry in RRGRAK
)

OUT

CENTRAL LOOP



delim negative? —yes→ HALT

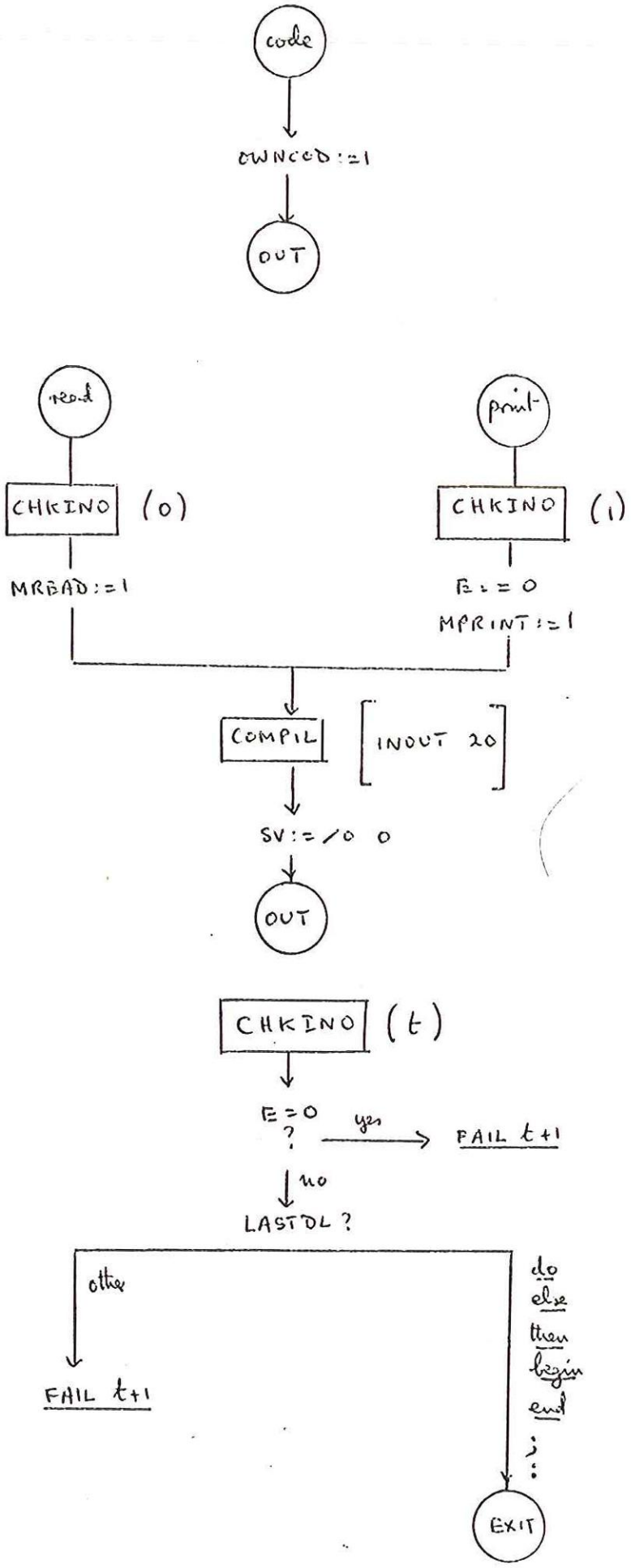
delim in table? —no→ FAIL 19

goto DELIM

```
'
(
)
* + - div ↑ /
,
:
;
< = > le ge ne
[
]
go to
if
for
end
print
read
begin
code
boolean
integer
real
array
switch
procedure
equiv impl or and not ≡ ⊃ ∨ ∧ ¬
then
else
do
:=
step until while
```

This is impossible to happen. Relic of early testing

code , read , print



precedes owncode declaration and is local to OUT

both local to OUT