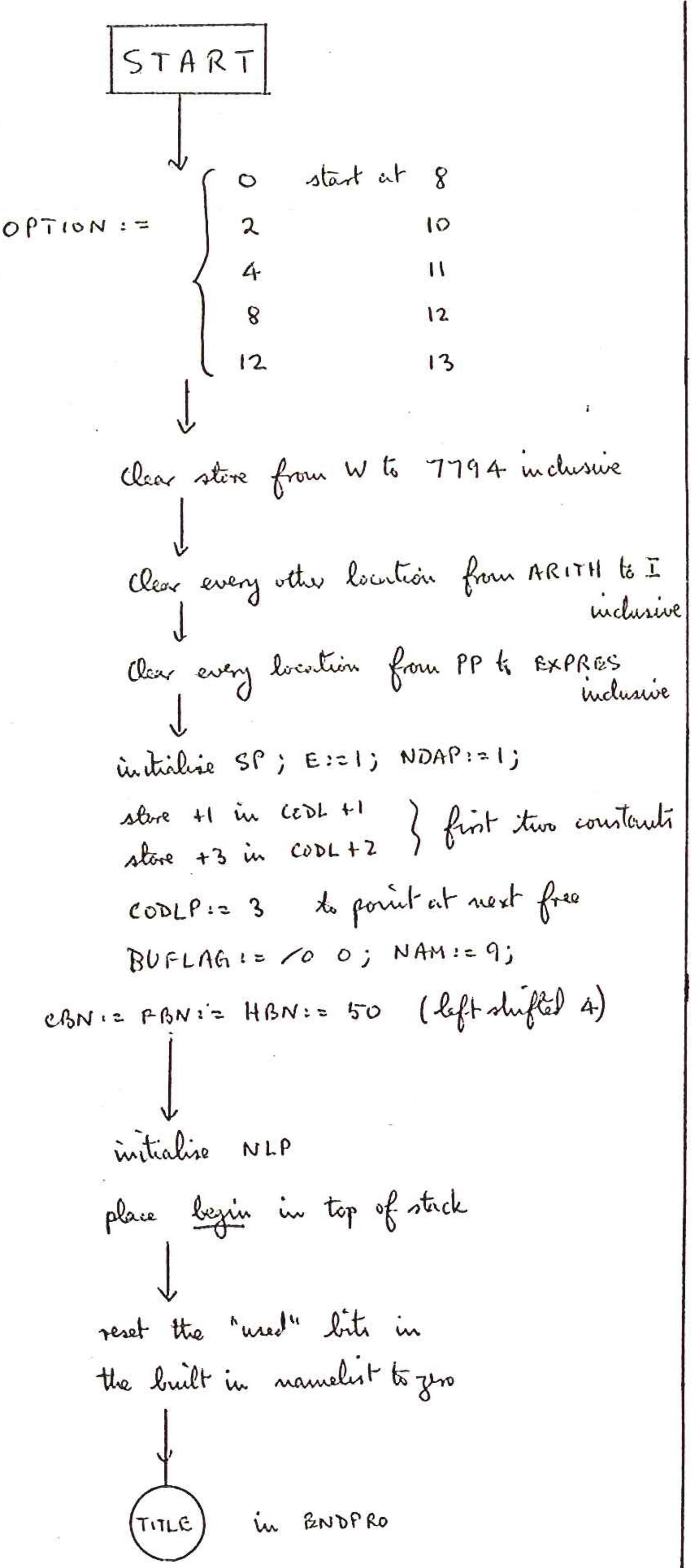
ELLIOTT 903 ALGOL

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 } first two constants
store +3 in CODL +2

CODLP:= 3 to point at next free

BUFLAG:= /0 0; NAM:=9;

CBN:= FBN:= HBN:= 50 (left shifted 4)

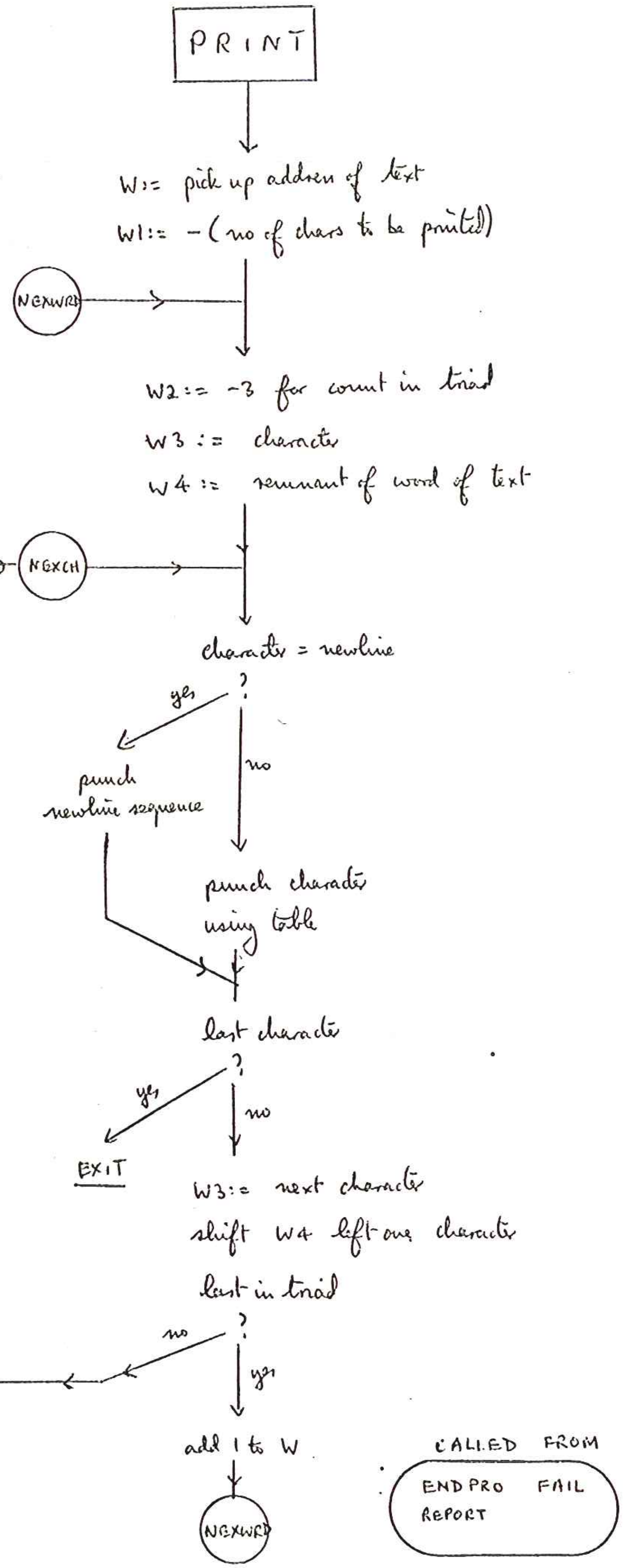
initialise NLP

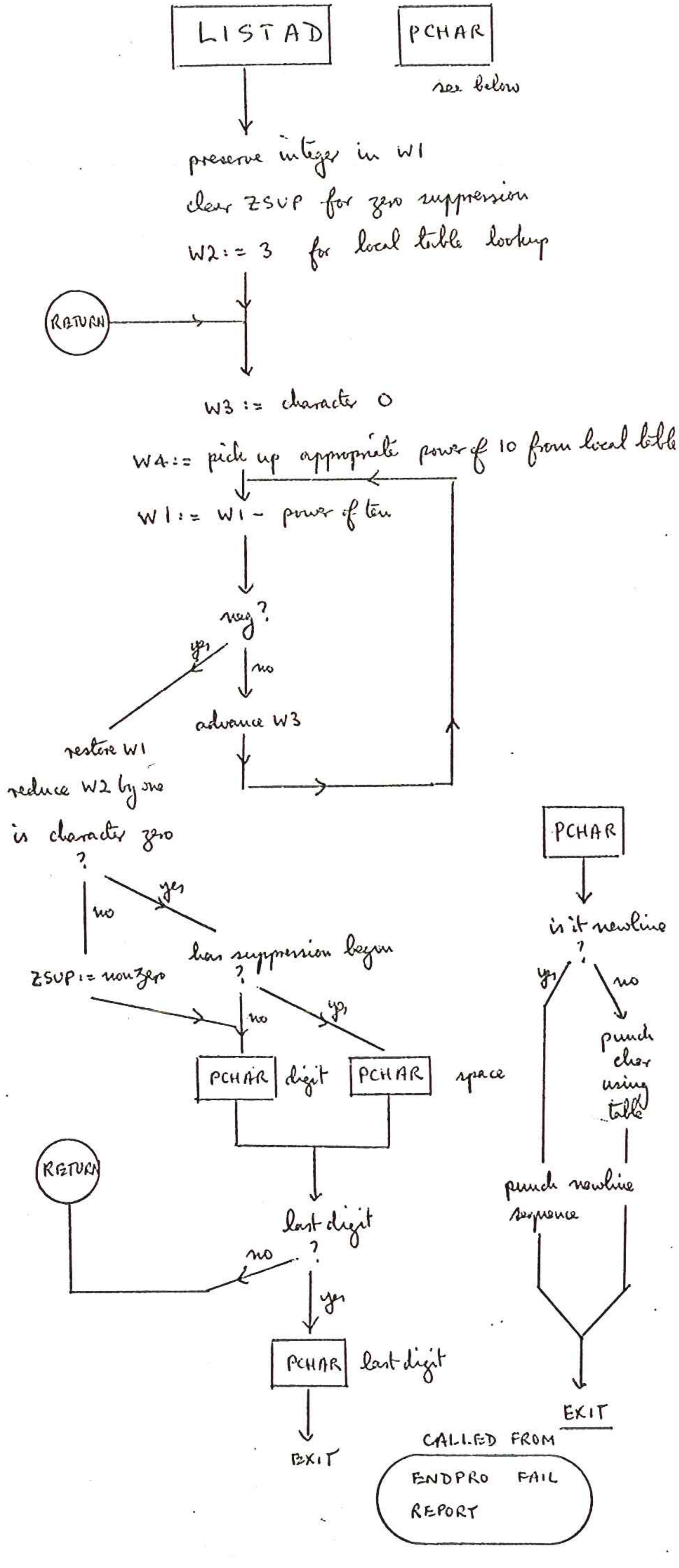
place <u>begin</u> in top of stack

reset the "used" bits in the built in namelist to zero

TITLE in ENDPRO

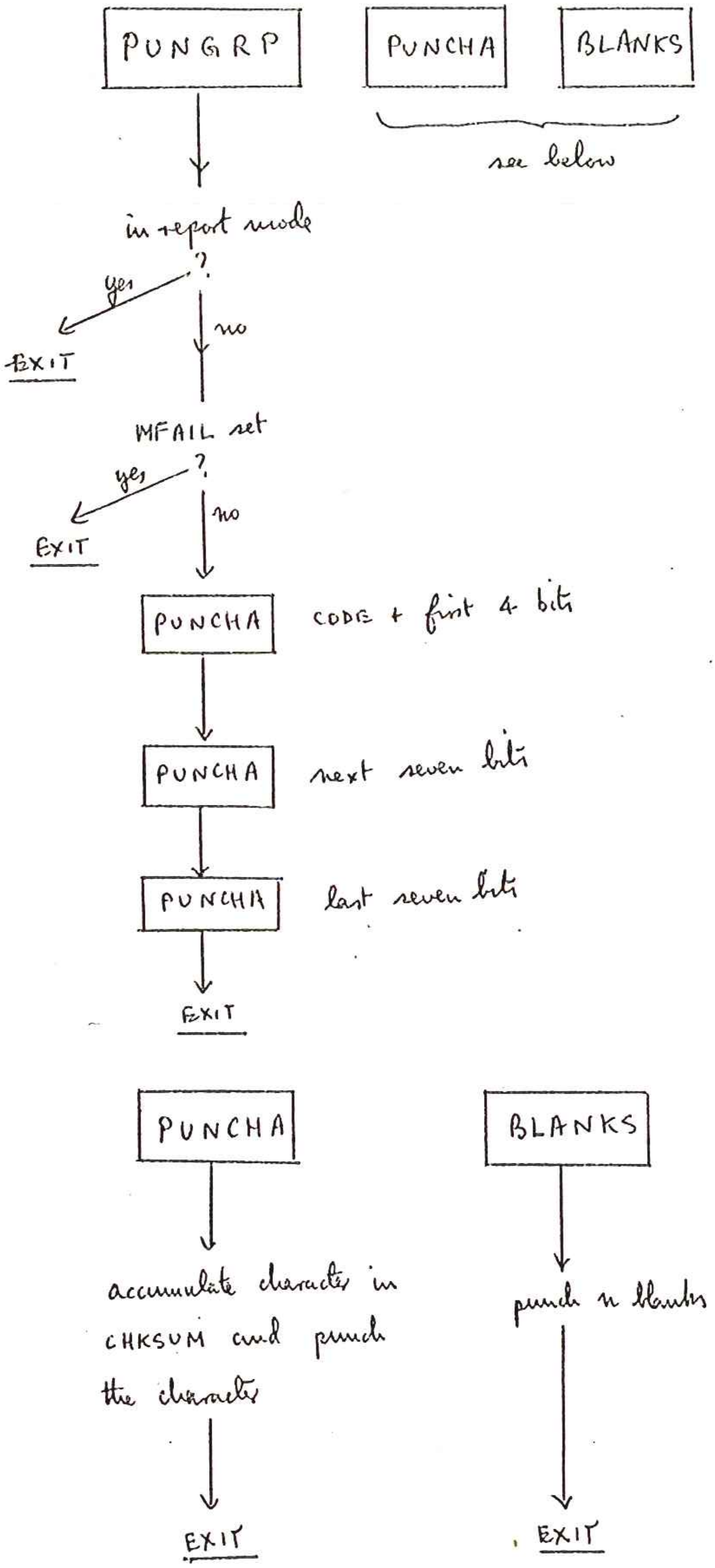
print text


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
ENDPRO FAIL
REPORT



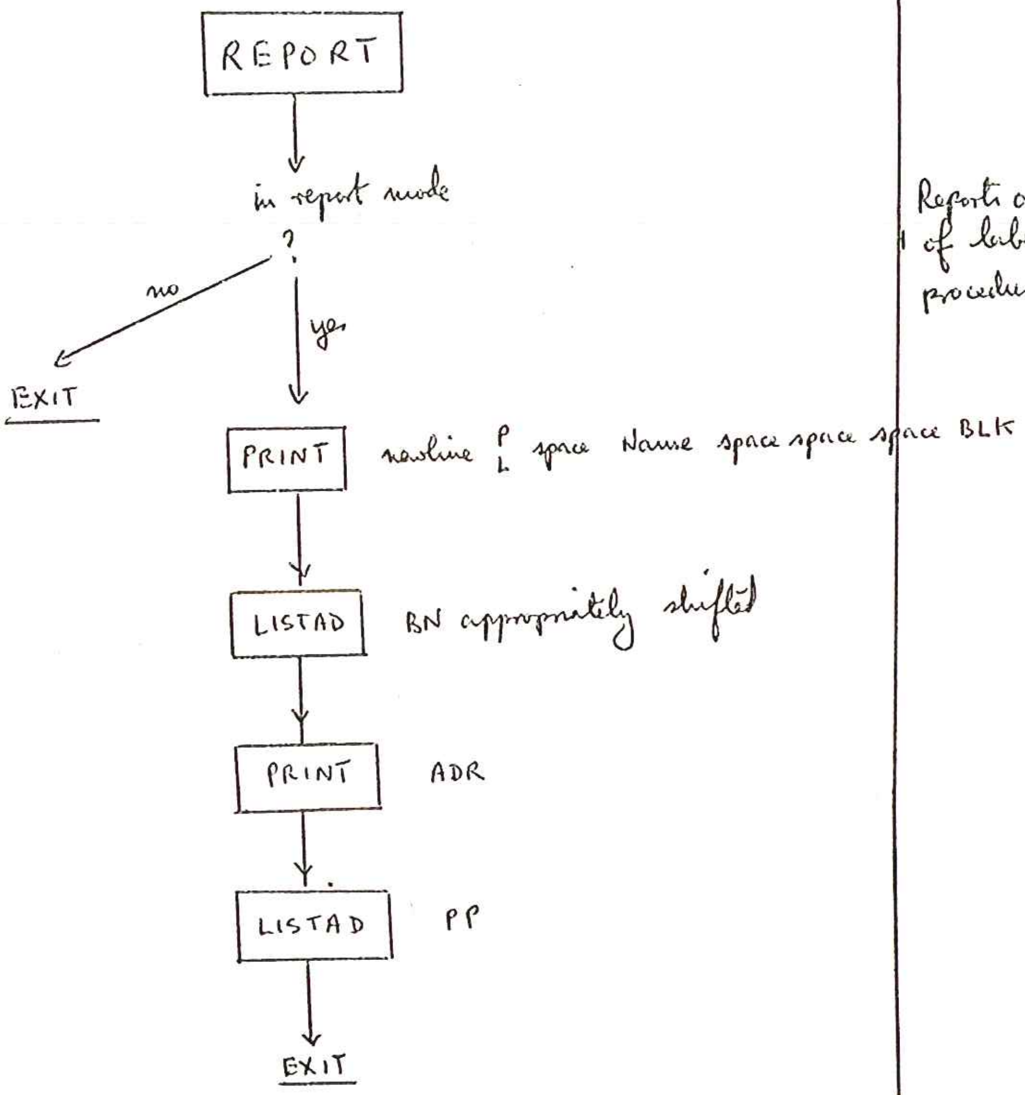
Convert binary integer and print it

punch a rel. bin. output word



CALLED FROM

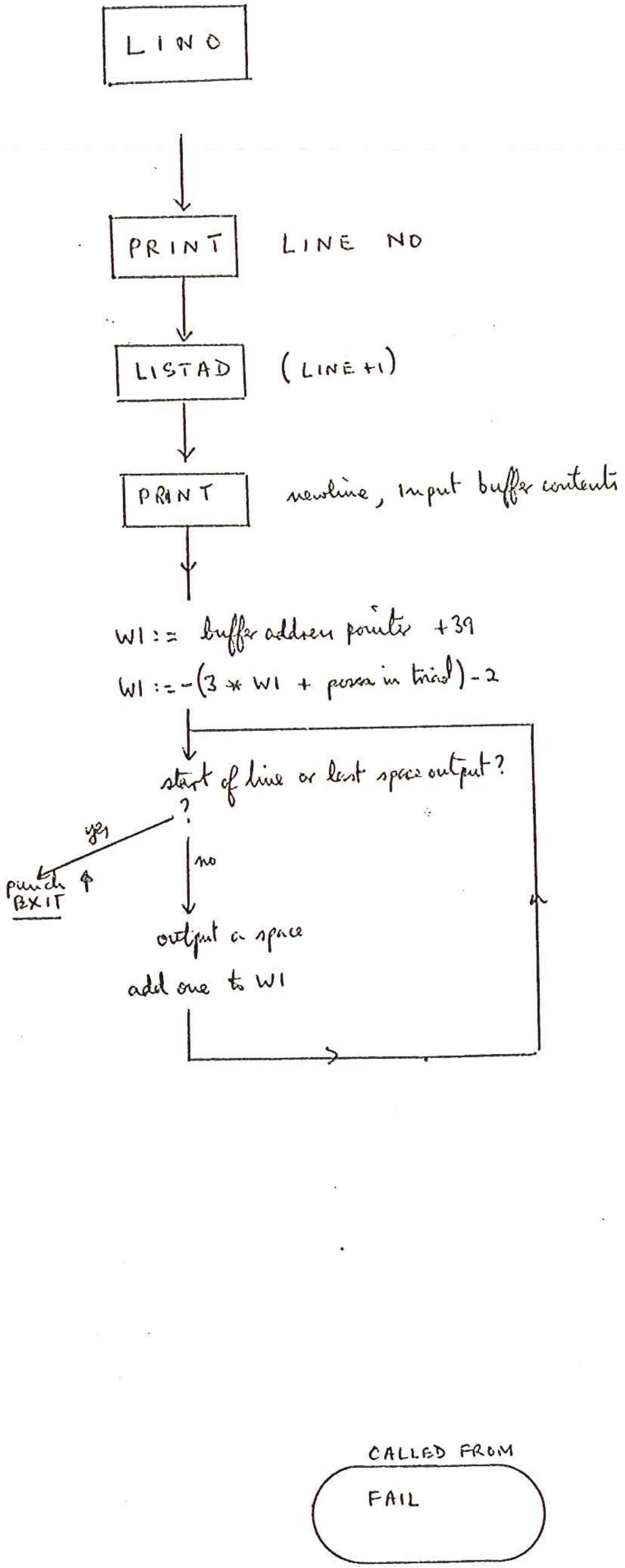
RRBRAK COMPIL
FOMPIL UPDATE
ENDPRO



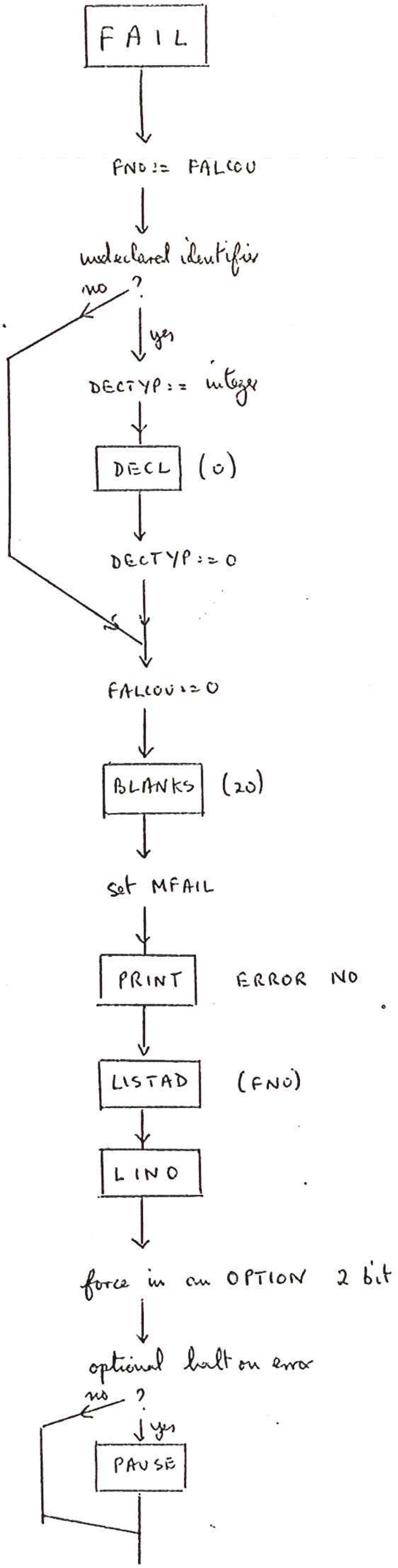
Reports occurrence of labels and procedures

CALLED FROM

PROCED
COLON


LINO
PRINT
LINE NO
LISTAD
(LINE +1)
PRINT
newline, input buffer contents
W1 := buffer address pointer +39
W1 := -(3 * W1 + posn in triad) - 2
start of line or last space output ?
?
yes
punch ↑
EXIT
no
output a space
add one to W1
CALLED FROM
FAIL

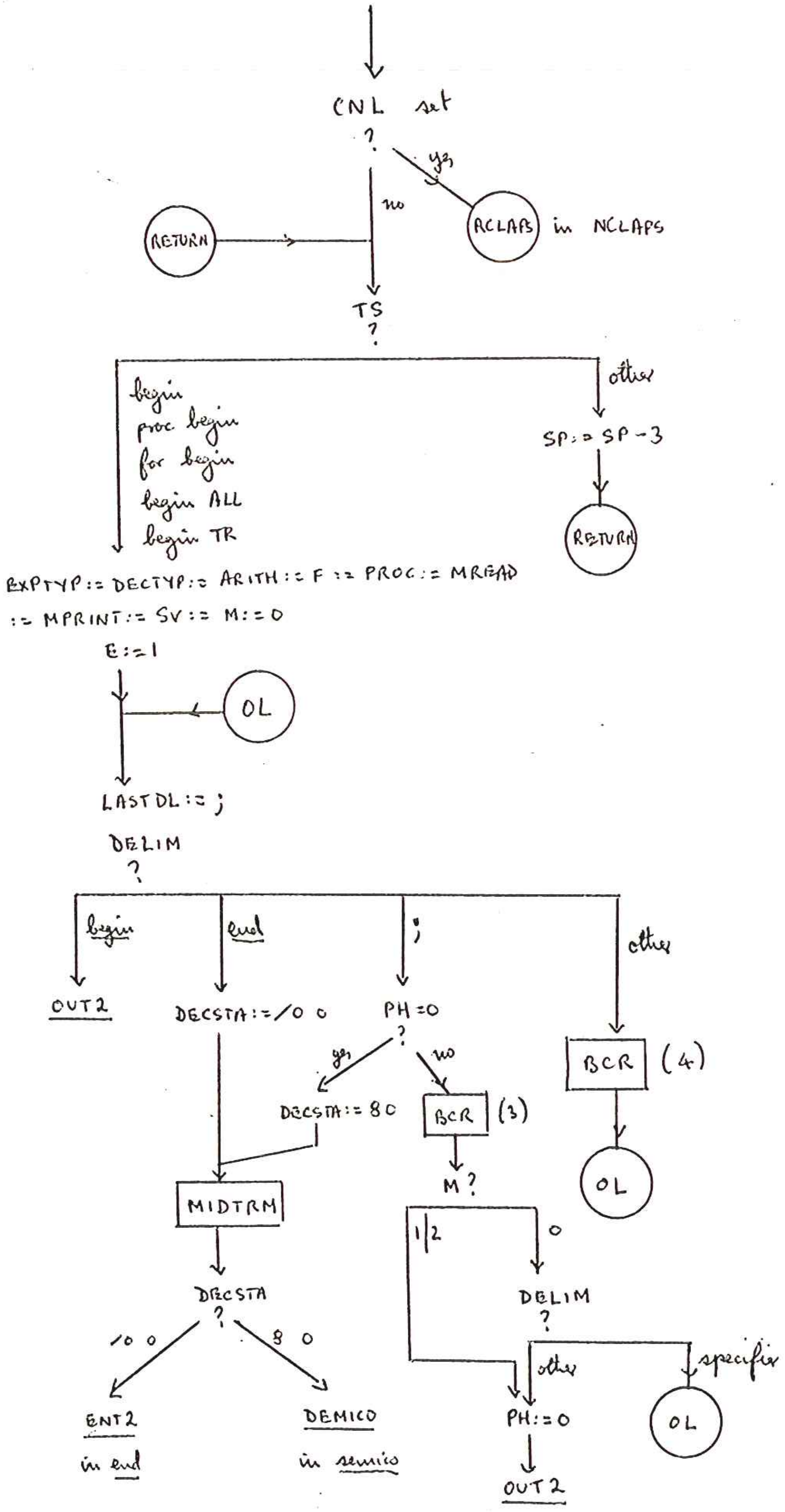
page 1 of 2



debugging facility

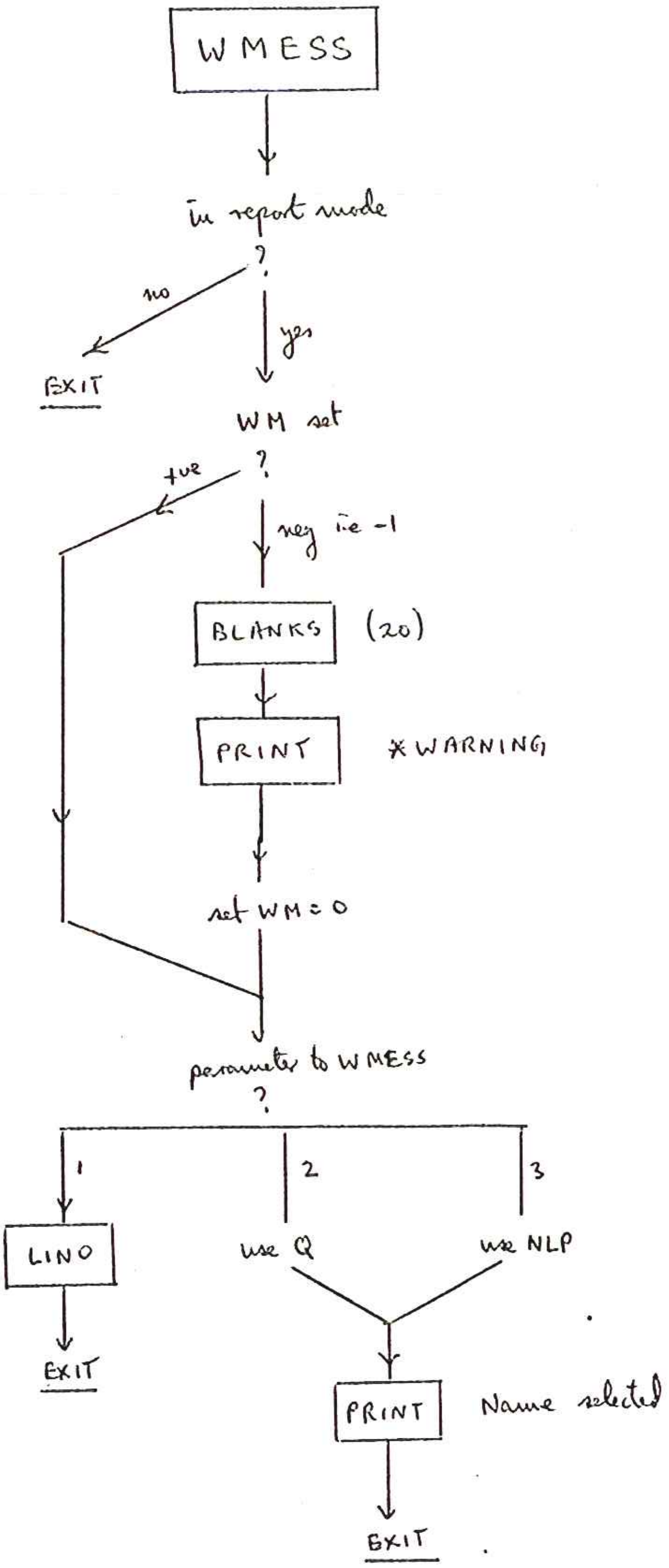
FAIL continued

page 2 of 2



for multiple fails during NCLAPS

MIDTRM is an entry in STATRM

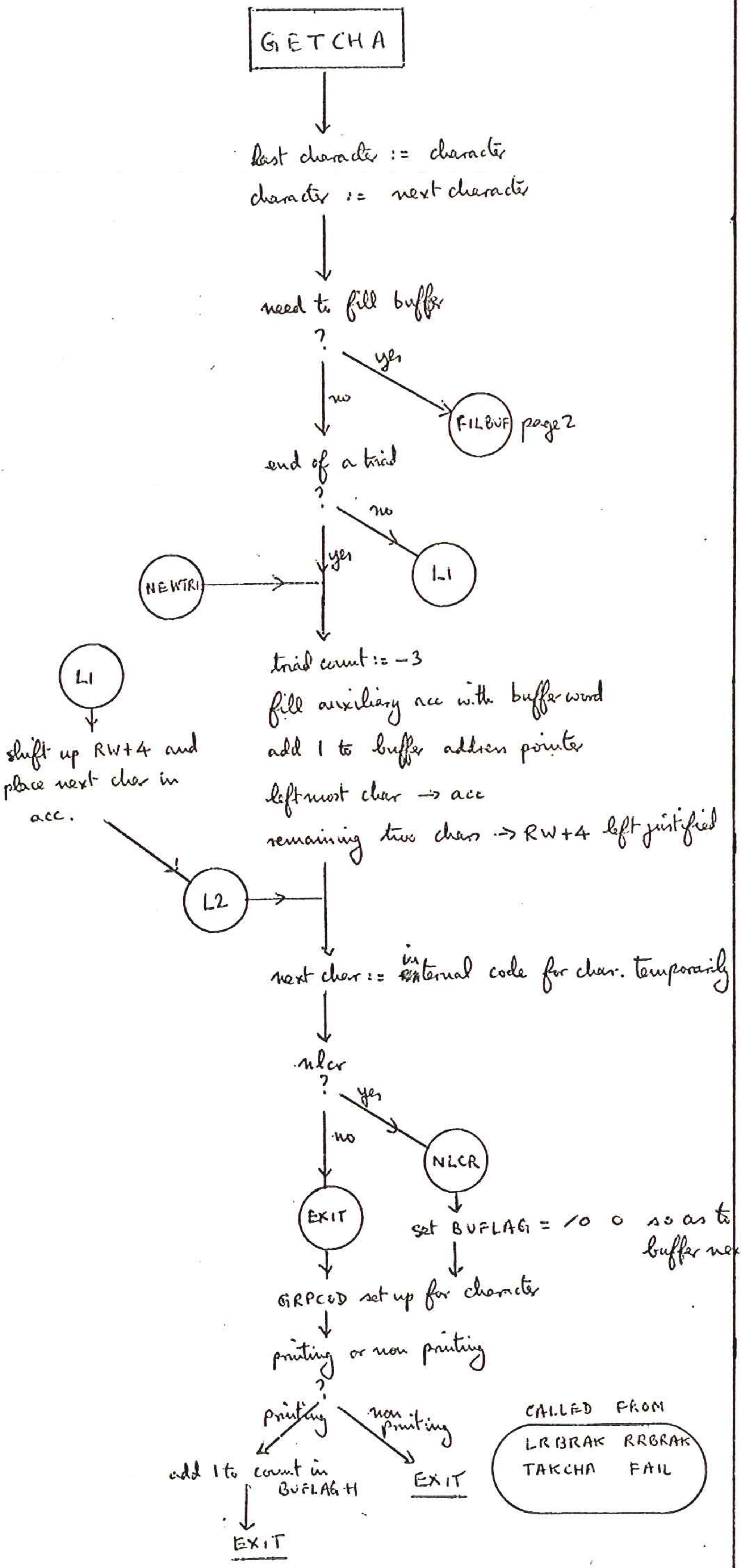


gives warning messages when in report mode

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

CALLED FROM

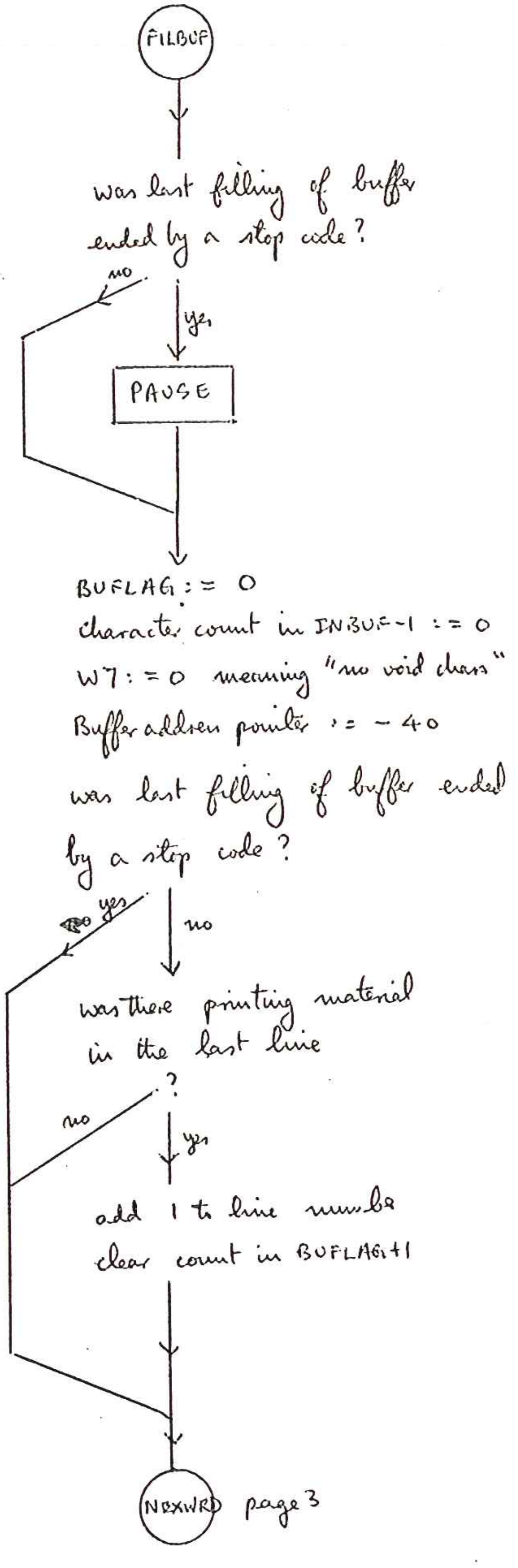
END FCLAPS
NCLAPS FAIL


GETCHA
last character := character
character := next character
need to fill buffer
?
yes
no
FILBUF
page 2
end of a triad
?
no
yes
NEWTRI
L1
triad count := -3
fill auxiliary acc with buffer word
add 1 to buffer address pointer
leftmost char → acc
remaining two chars → RW+4 left justified
L1
shift up RW+4 and place next char in acc.
L2
next char := internal code for char. temporarily
nlcr
?
yes
no
NLCR
EXIT
set BUFLAG = /0 0 so as to fill the buffer next time
GRPCOD set up for character
printing or non printing
?
printing
non printing
add 1 to count in BUFLAG+1
EXIT
EXIT
CALLED FROM
LRBRAK RRBRAK
TAKCHA FAIL

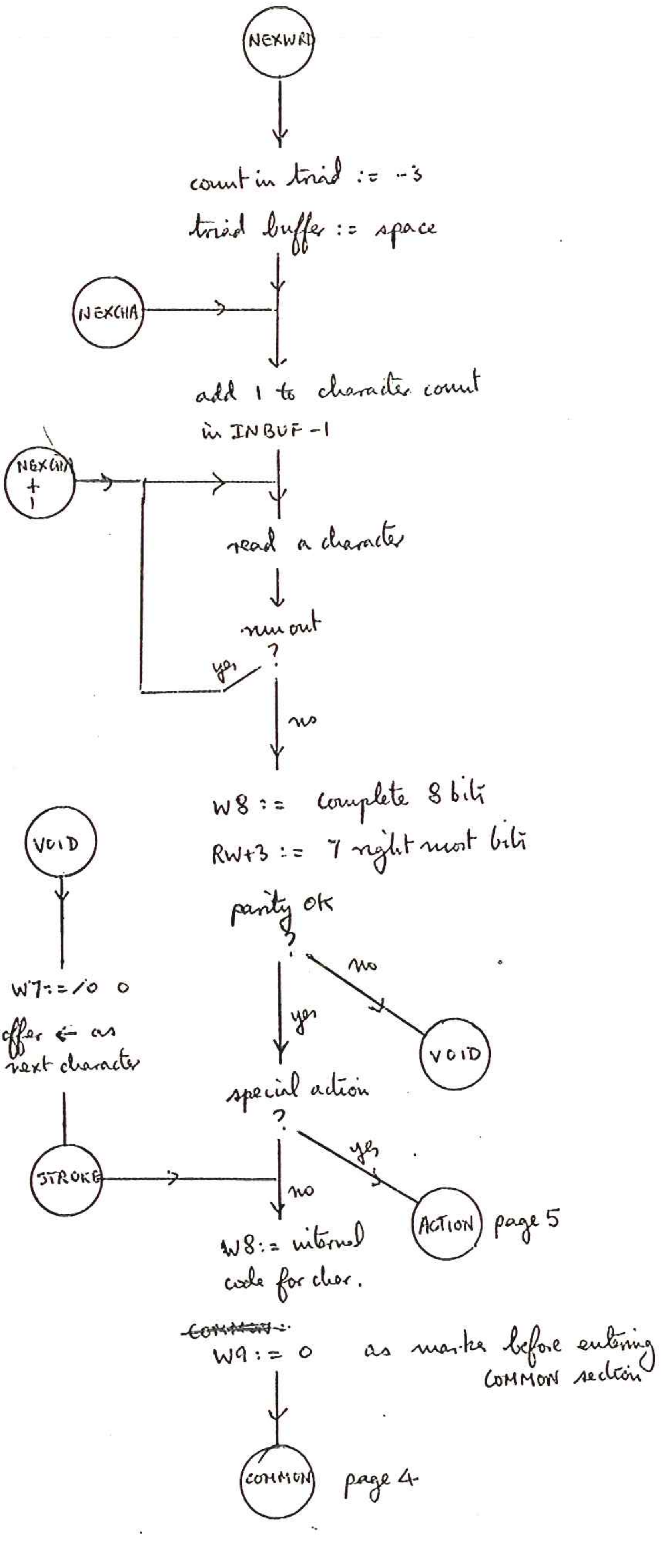
page 1 of 5

supply next character

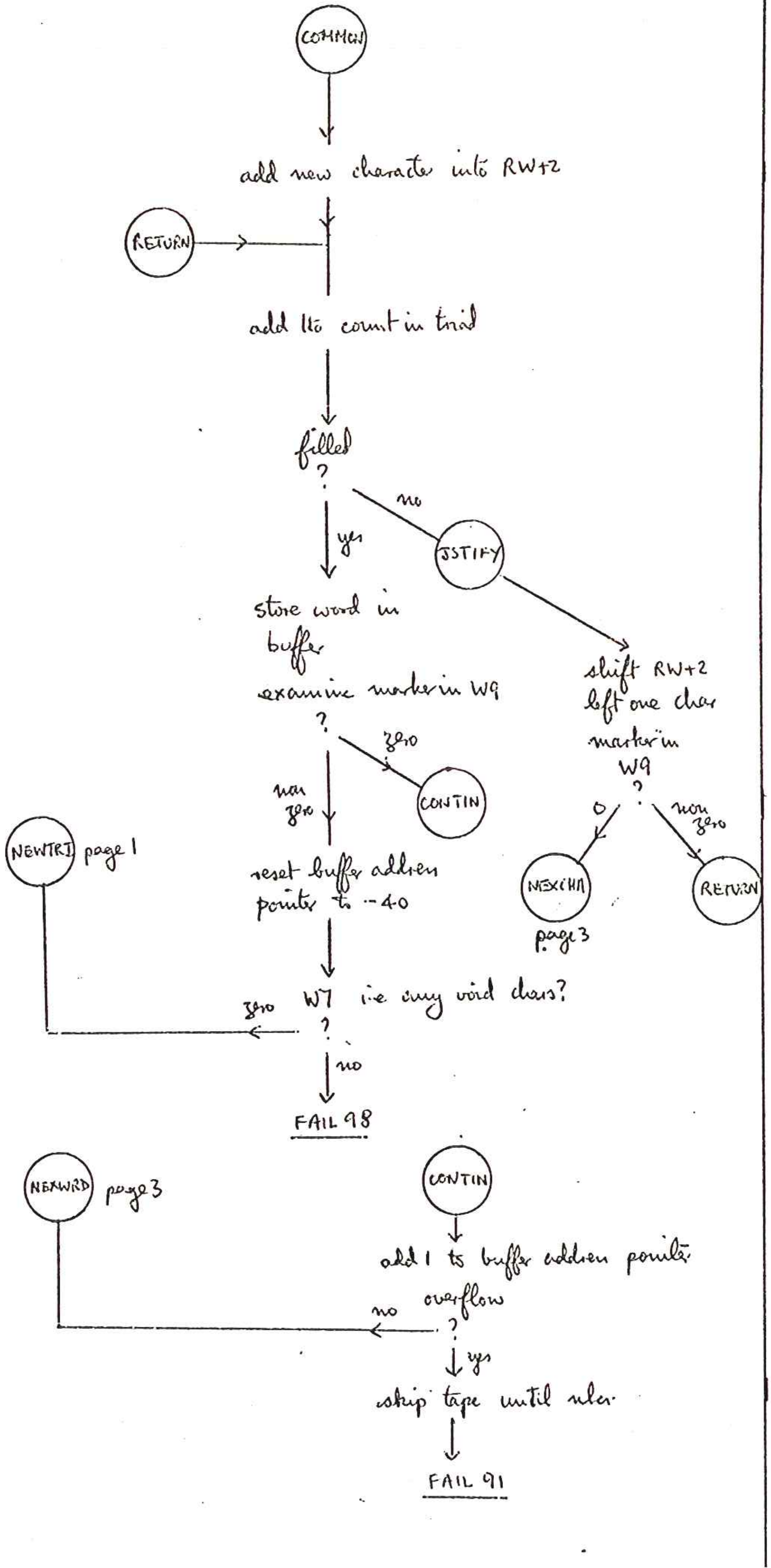
GETCHA


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

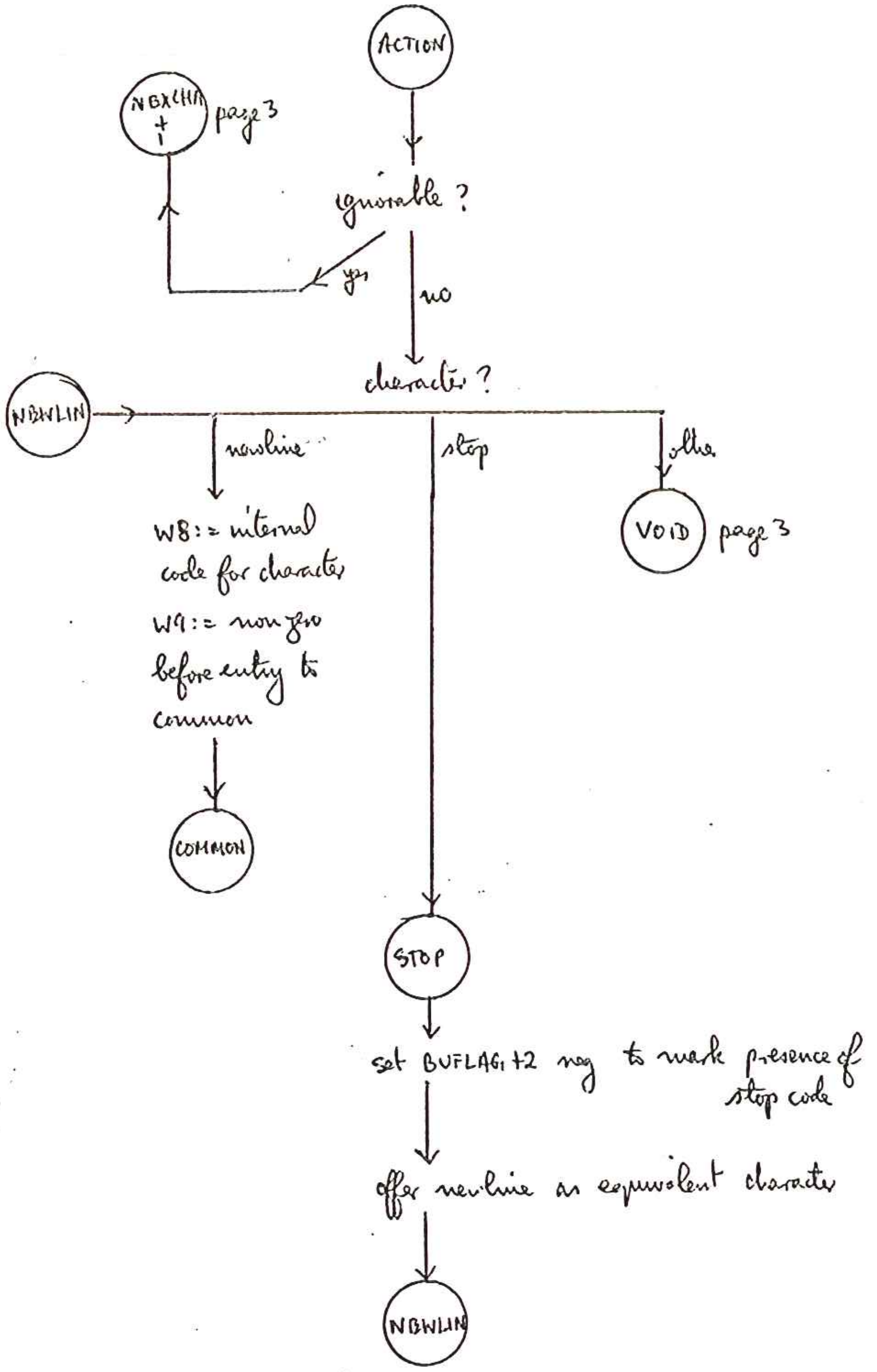
GETCHA continued



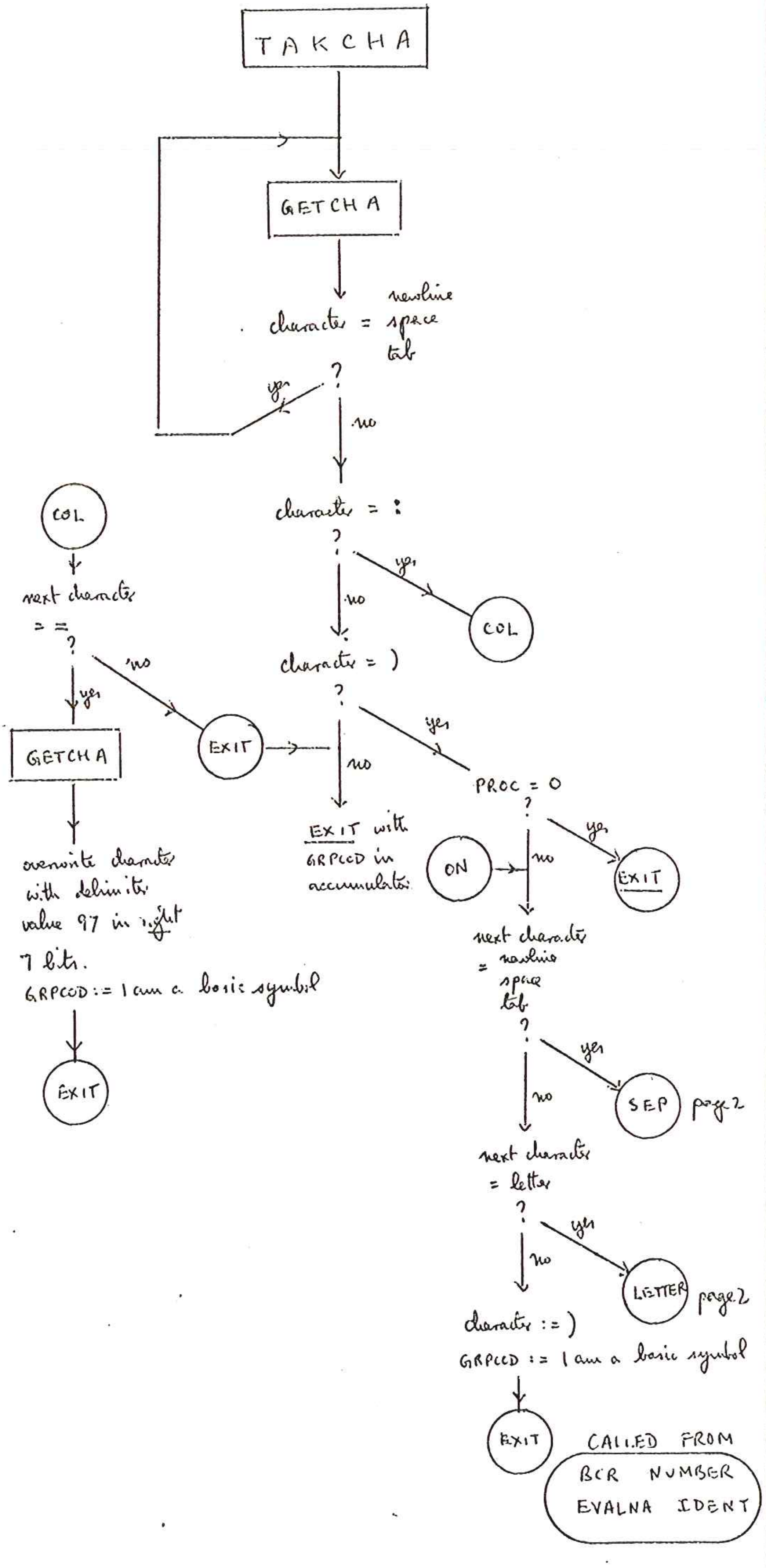
GETCHA continued



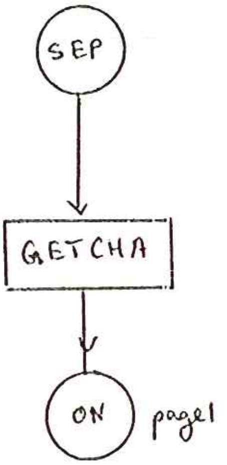
GETCHA continued

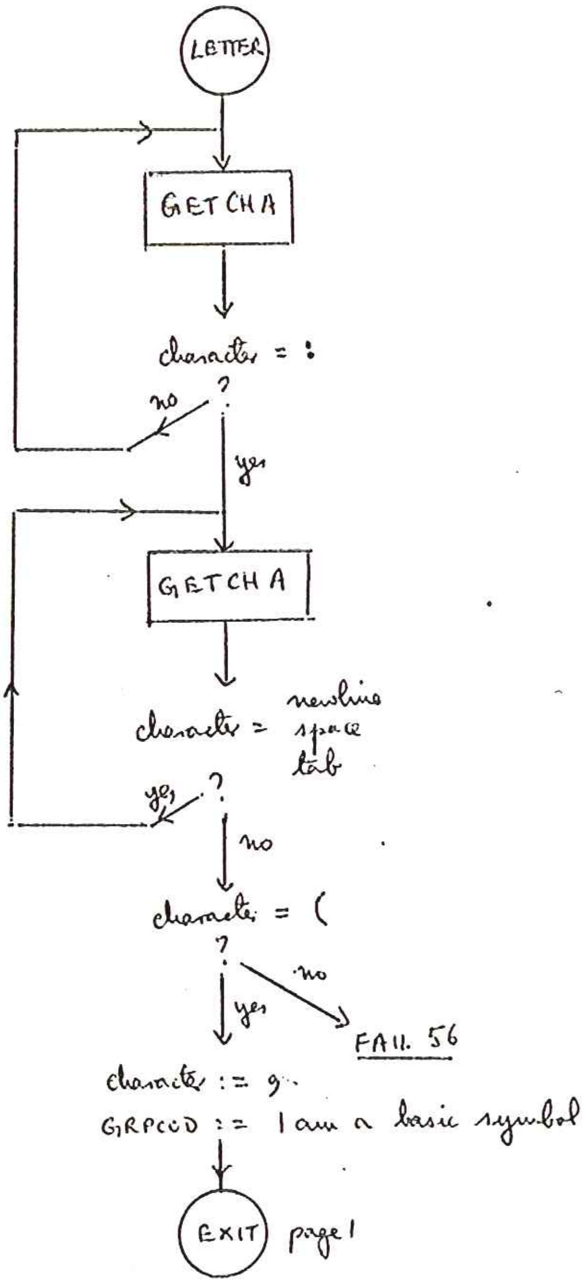

ACTION
NBXCHA +1
page 3
ignorable ?
yes
no
character ?
NBWLIN
newline
stop
other
VOID
page 3
W8:= internal code for character
W9:= non zero before entry to common
COMMON
STOP
set BUFLAG+2 neg to mark presence of stop code
offer newline as equivalent character
NBWLIN

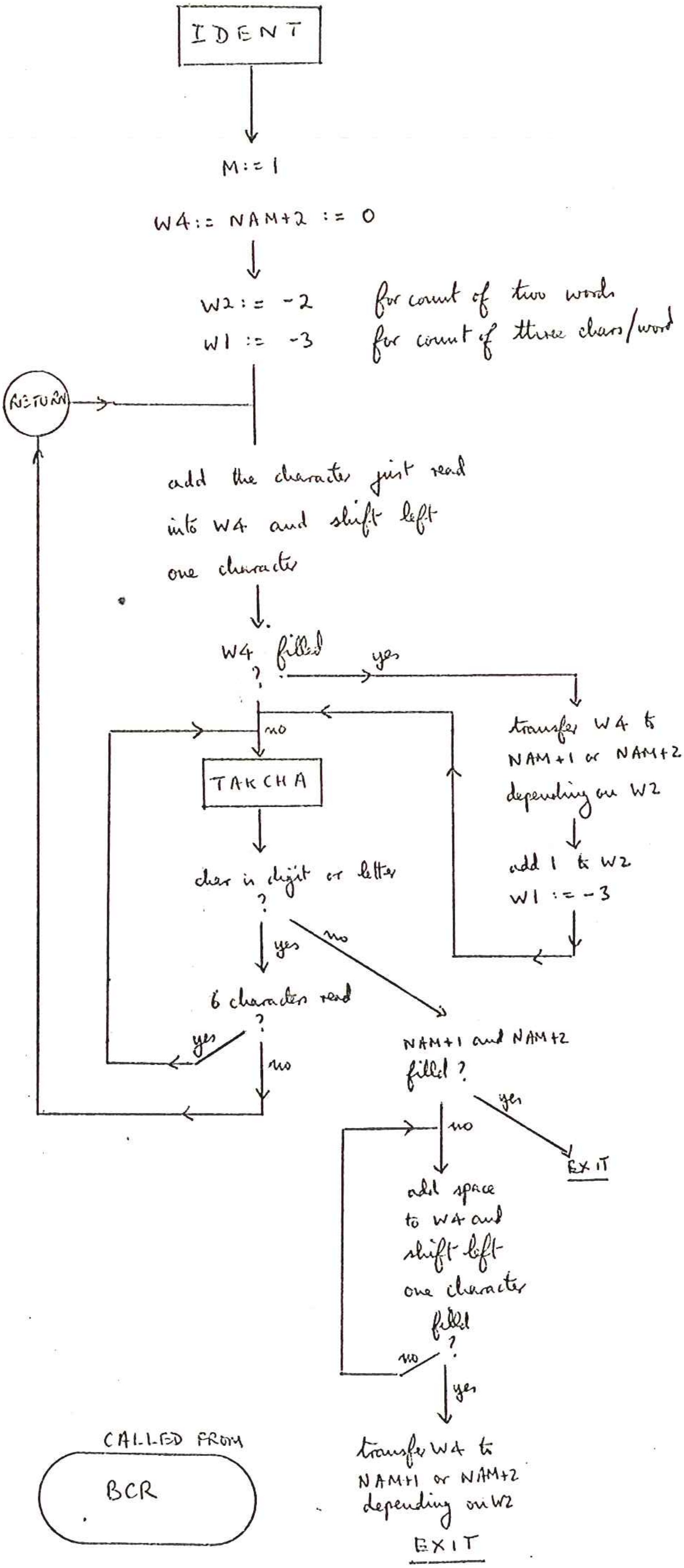
page1 of 2


TAKCHA
GETCHA
character = newline space tab
?
yes
no
character = :
?
yes
no
COL
character =)
?
yes
no
EXIT with GRPCOD in accumulator
PROC = 0
?
yes
no
EXIT
ON
next character = newline space tab
?
yes
no
SEP
page 2
next character = letter
?
yes
no
LETTER
page 2
character :=)
GRPCOD := 1 am a basic symbol
EXIT
CALLED FROM
BCR NUMBER
EVALNA IDENT
COL
next character = =
?
yes
no
EXIT
GETCHA
overwrite character with delimiter value 97 in right 7 bits.
GRPCOD := 1 am a basic symbol
EXIT

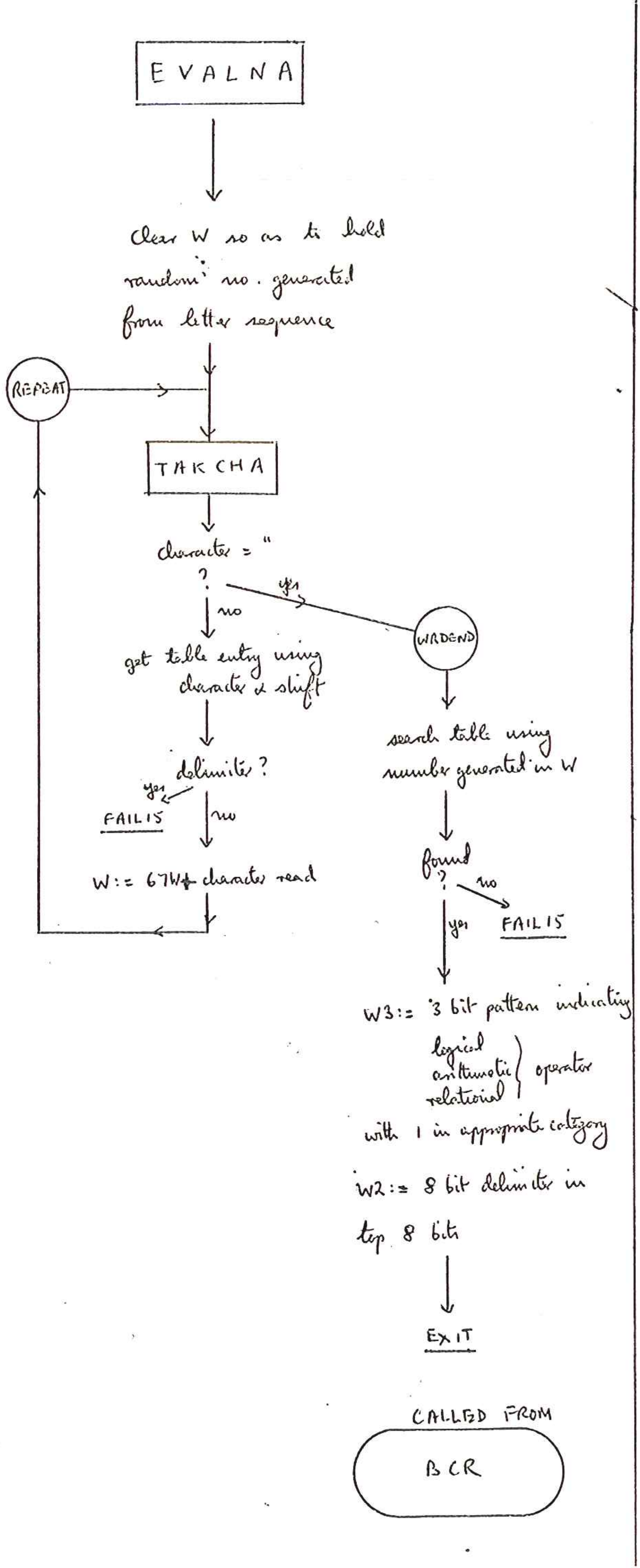
TAKCHA continued



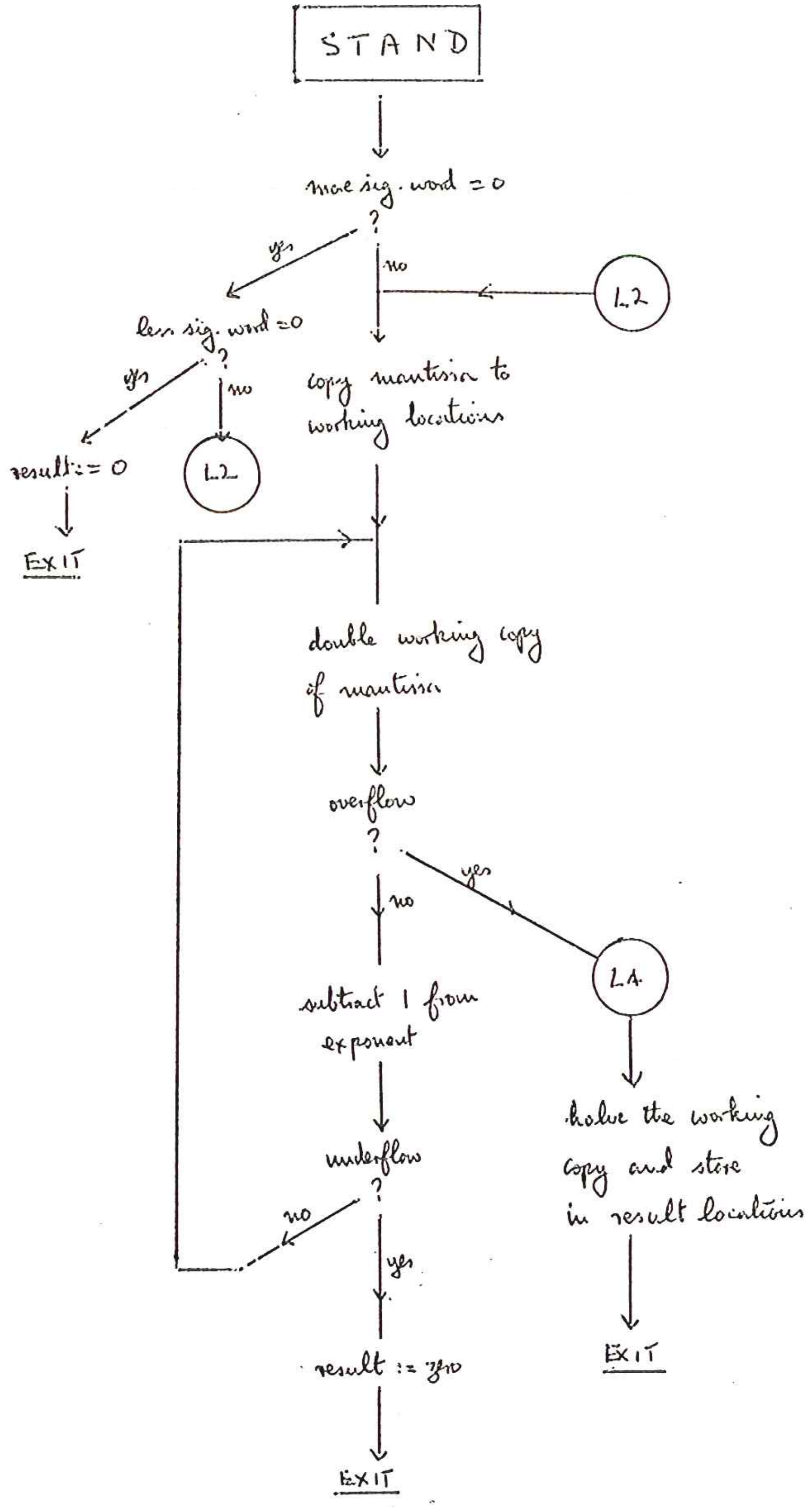



IDENT
M:=1
W4:= NAM+2 := 0
W2:= -2
for count of two words
W1 := -3
for count of three chars/word
RETURN
add the character just read into W4 and shift left one character
W4 filled ?
yes
no
transfer W4 to NAM+1 or NAM+2 depending on W2
add 1 to W2
W1 := -3
TAKCHA
char is digit or letter ?
yes
no
6 characters read ?
yes
no
NAM+1 and NAM+2 filled ?
yes
EXIT
no
add space to W4 and shift left one character
filled ?
no
yes
transfer W4 to NAM+1 or NAM+2 depending on W2
EXIT
CALLED FROM
BCR

select up to first 6 chars of identifier



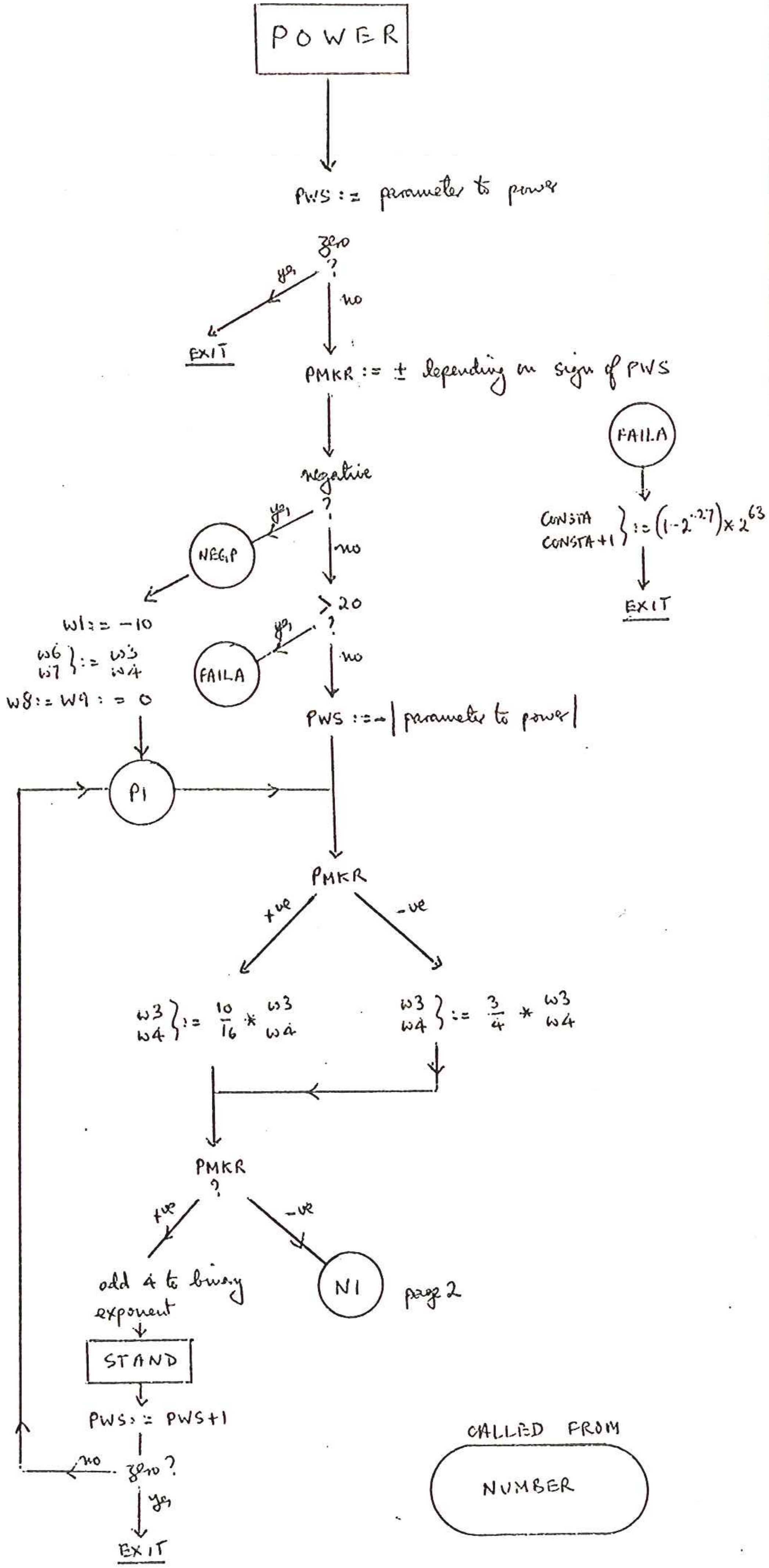
convert "underlined"
word to
delimiter value



standardise contents of W3W4W5

CALLED FROM

NUMBER

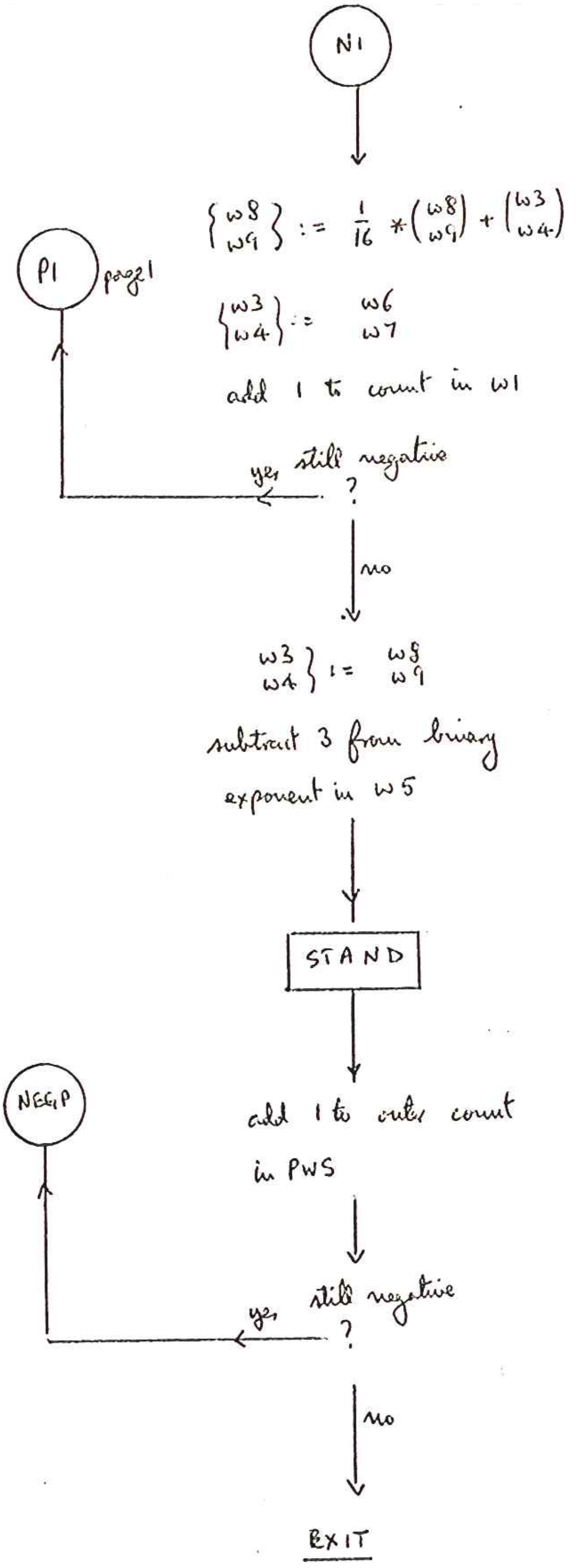


page 1 of 2

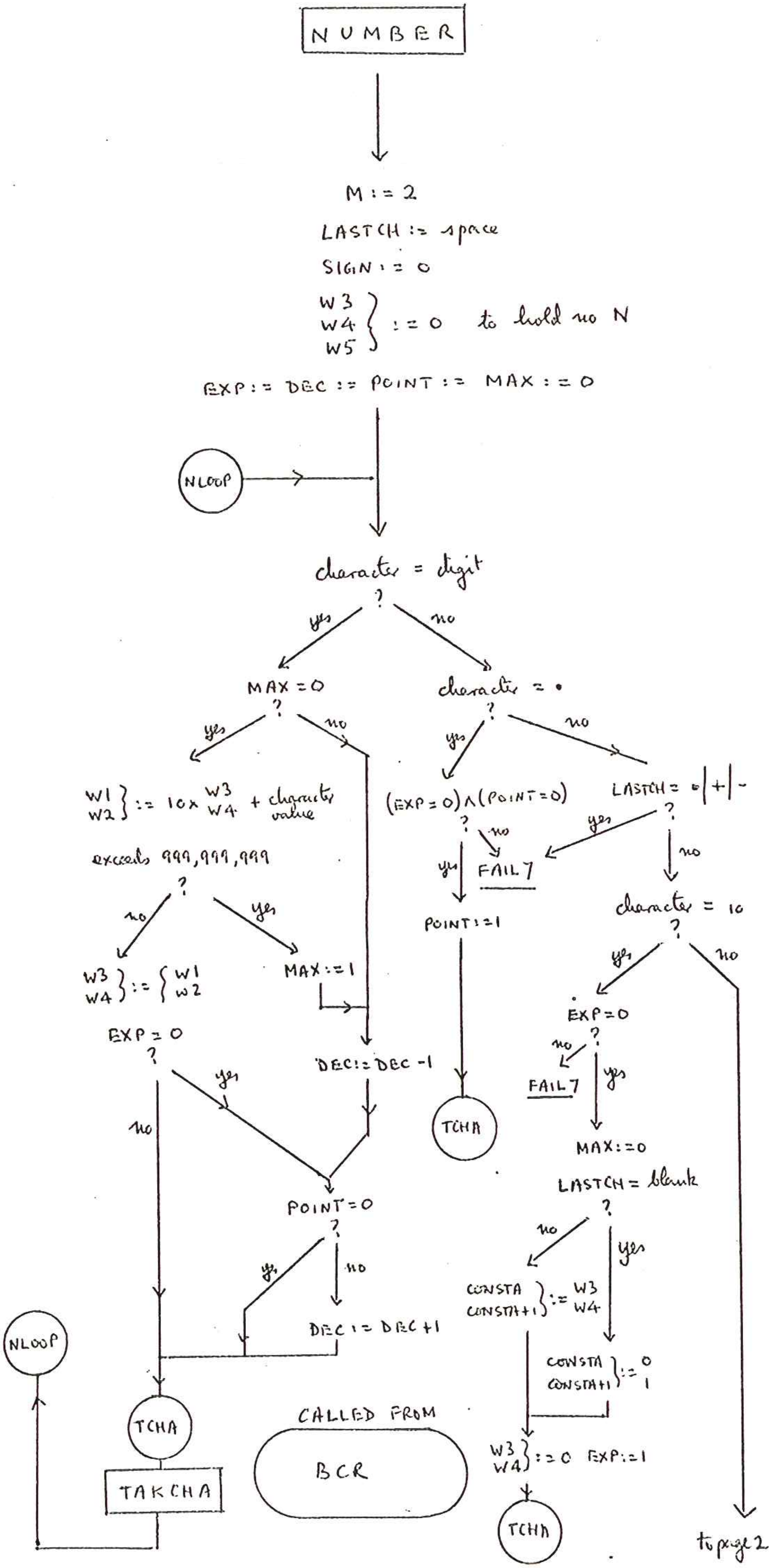
raise contents of w3 w4 w5 to power of ten given in Acc.

POWER continued

page 2 of 2


N1
$\begin{Bmatrix} w8 \\ w9 \end{Bmatrix} := \frac{1}{16} * \begin{pmatrix} w8 \\ w9 \end{pmatrix} + \begin{pmatrix} w3 \\ w4 \end{pmatrix}$
$\begin{Bmatrix} w3 \\ w4 \end{Bmatrix} := \begin{matrix} w6 \\ w7 \end{matrix}$
add 1 to count in w1
P1
page 1
yes
still negative
?
no
$\begin{matrix} w3 \\ w4 \end{matrix}\Big\} := \begin{matrix} w8 \\ w9 \end{matrix}$
subtract 3 from binary
exponent in w5
STAND
NEGP
add 1 to outer count
in PWS
yes
still negative
?
no
EXIT

This is a programmed multiplication by the constant 0.000110011.... to 40 binary places. This is $\frac{1}{10}$.

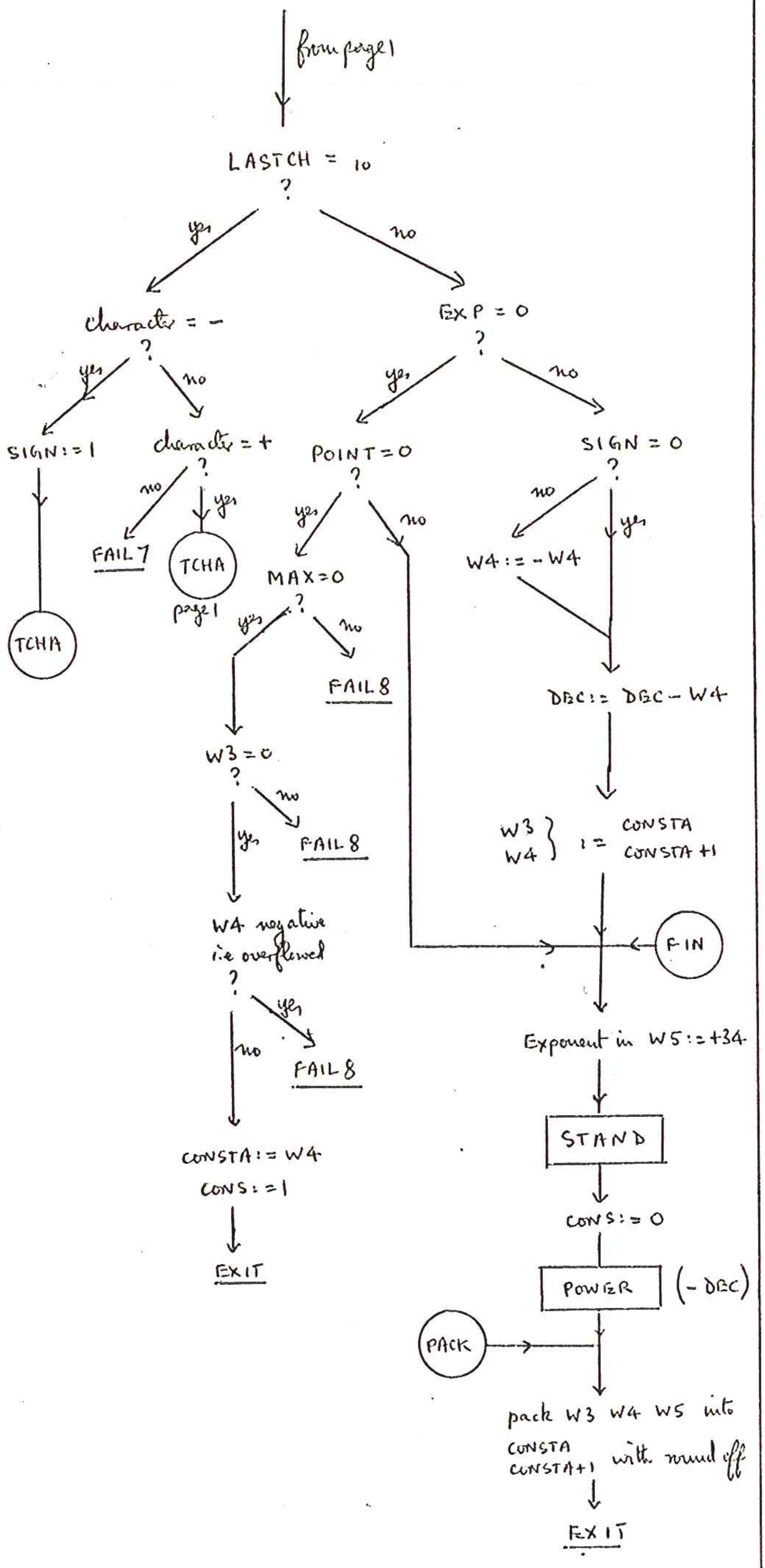


page 1 of 2

convert input no. to floating binary.

NUMBER continued

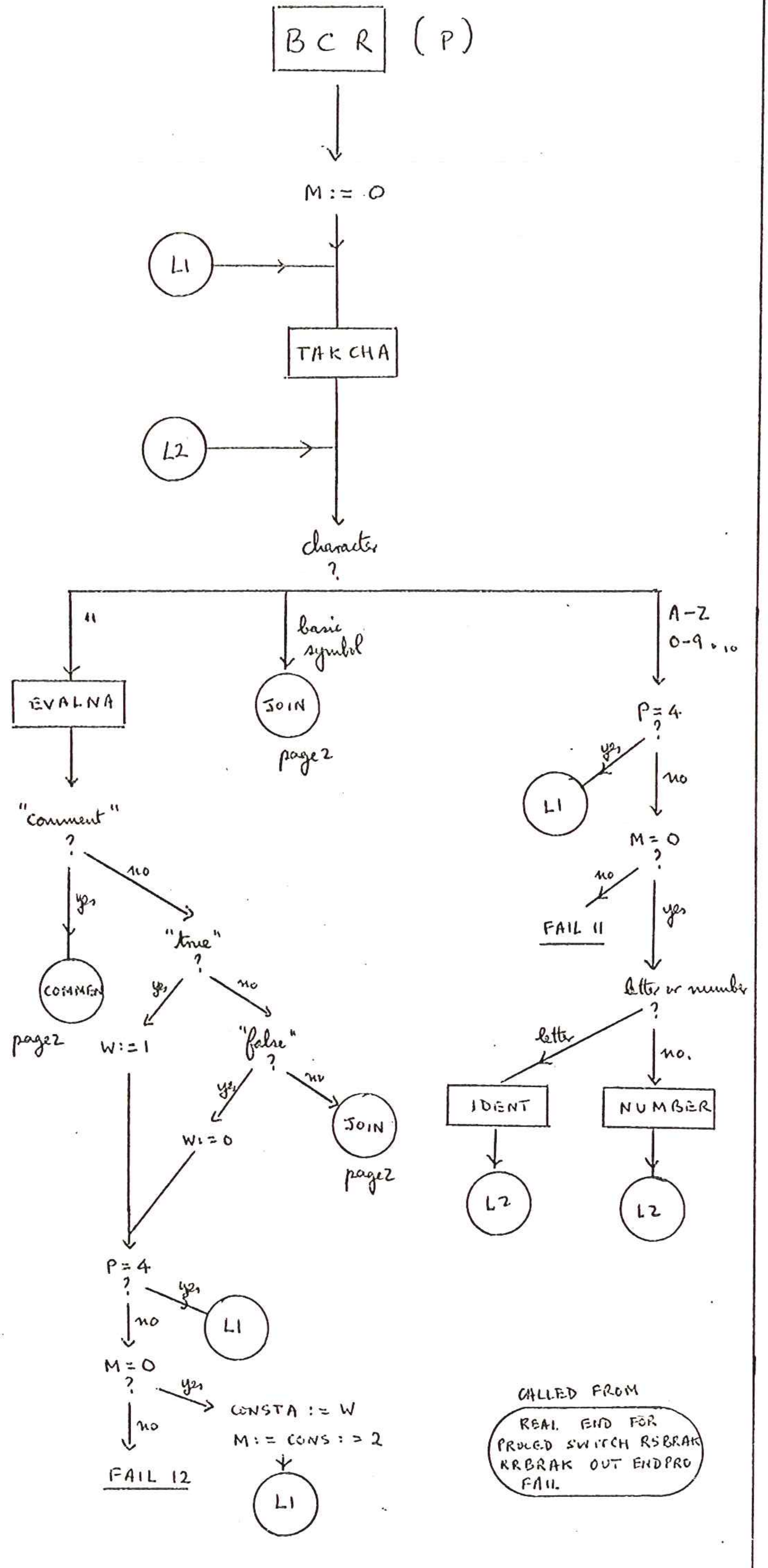
page 2 of 2



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

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

page 1 of 2.

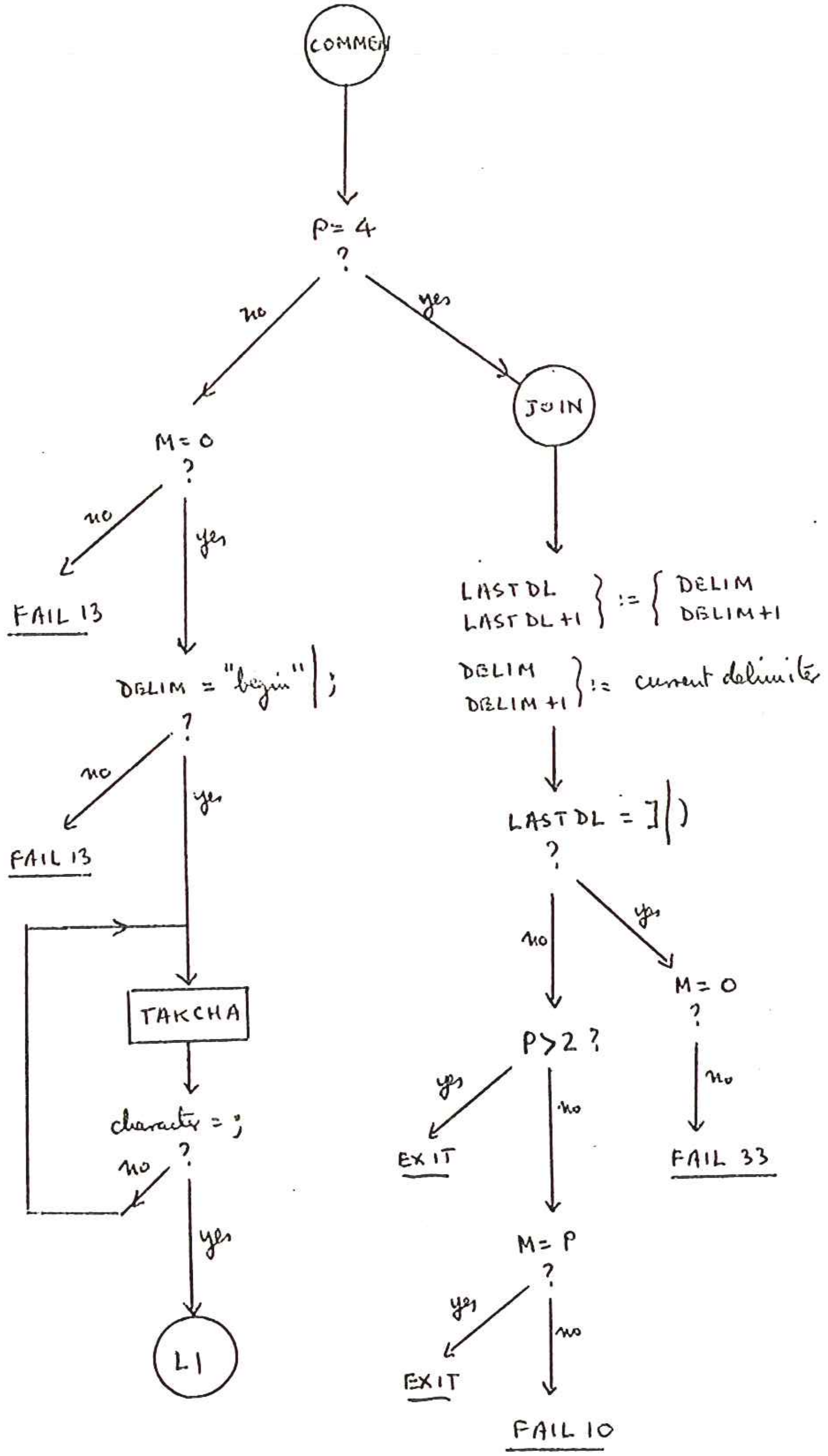


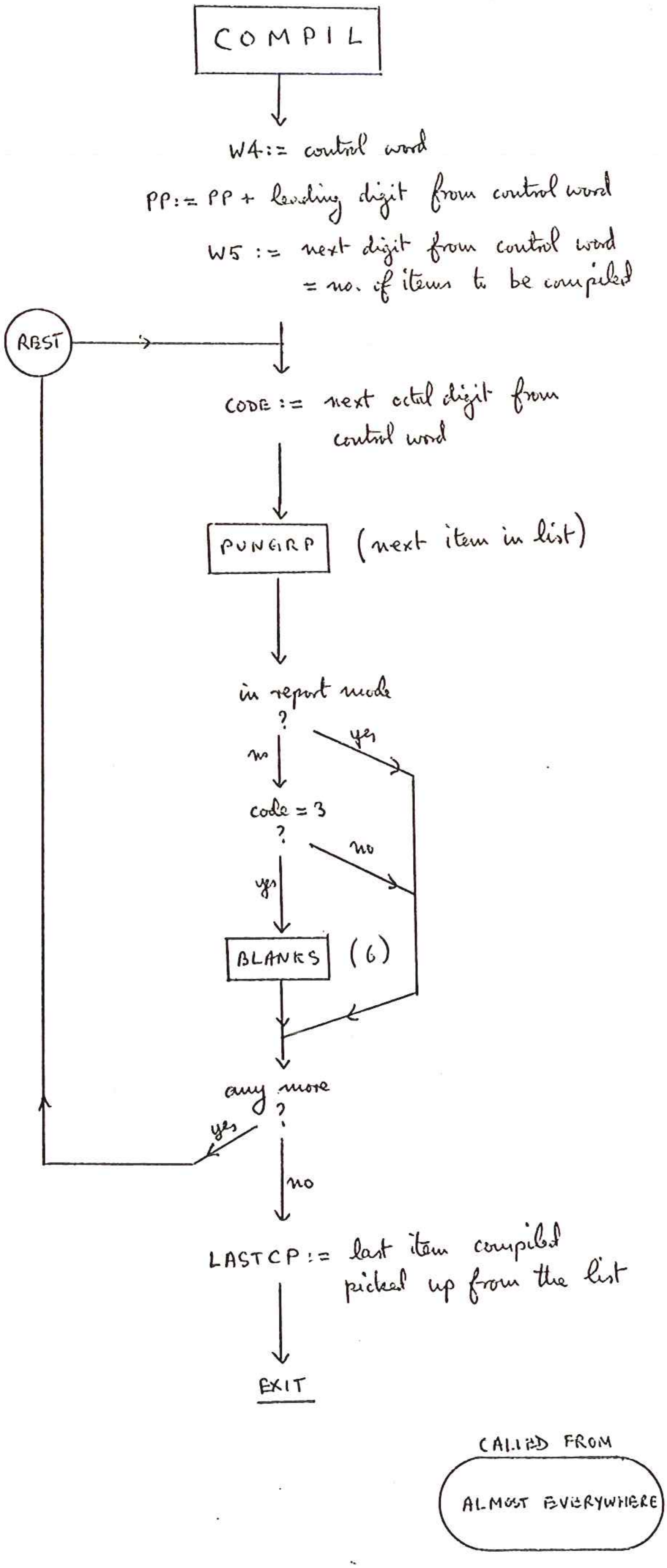
CALLED FROM

REAL END FOR
PROCED SWITCH RSBRAK
RRBRAK OUT ENDPRO
FAIL

BCR continued

page 2 of 2



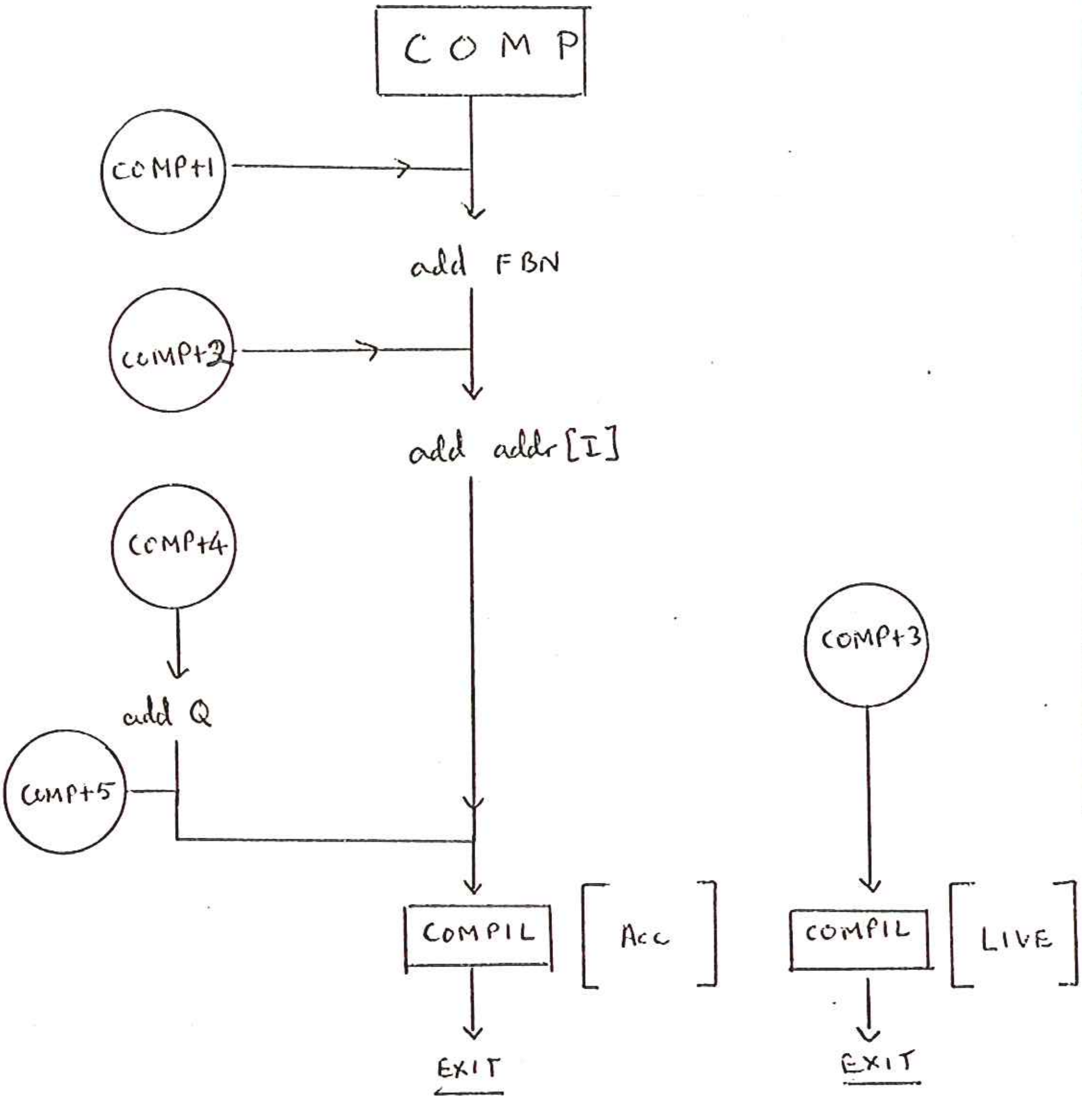

COMPIL
W4:= control word
PP:= PP + leading digit from control word
W5 := next digit from control word
= no. of items to be compiled
RBST
CODE := next octal digit from control word
PUNGRP
(next item in list)
in report mode ?
yes
no
code = 3 ?
no
yes
BLANKS
(6)
any more ?
yes
no
LASTCP := last item compiled picked up from the list
EXIT
CALLED FROM
ALMOST EVERYWHERE

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

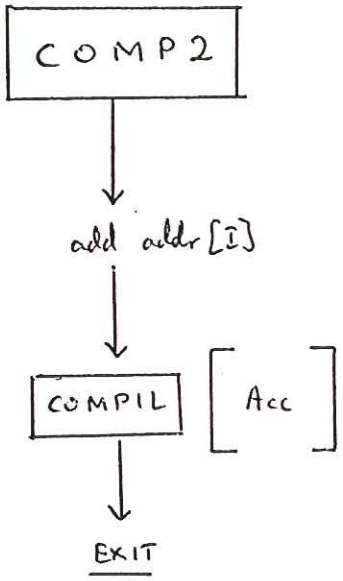
PP:= PP + O_1

O_2 = no of words to compile

$O_3 - O_6$ = codes



Compiles with SIR loader code 1

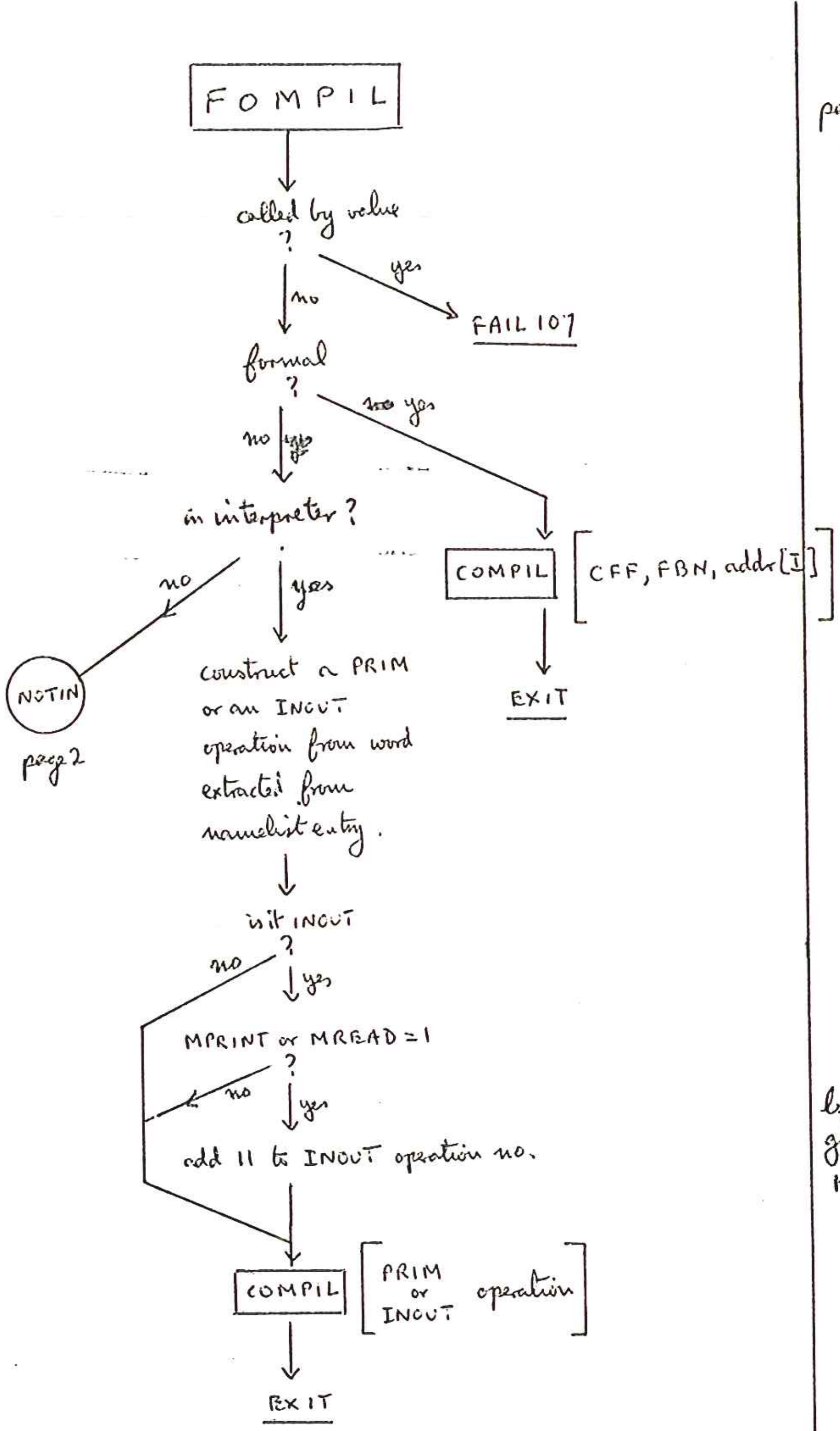


This is with SIR loader code 2

COMP2
CALLED FROM
COMPIL BRANCH
TAKEN

COMP
CALLED FROM
ALMOST EVERYWHERE

page 1 of 2

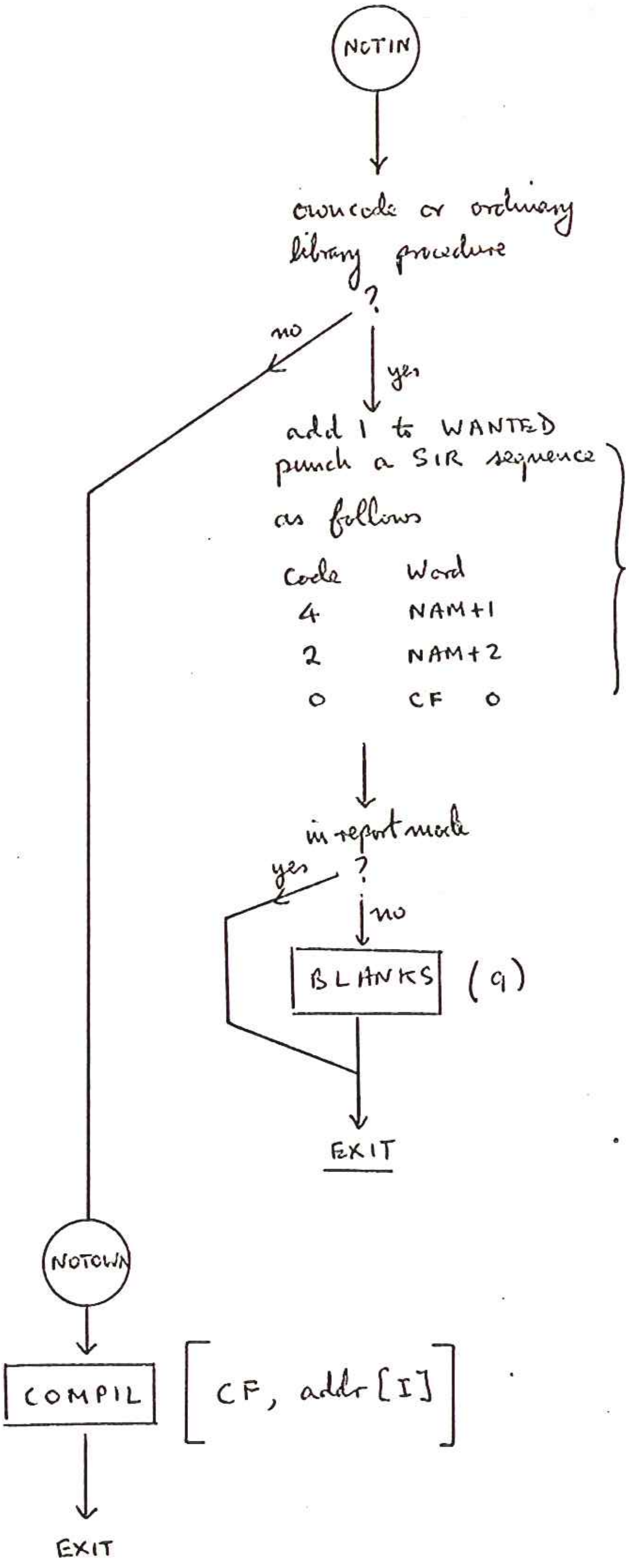


local not global if MREAD or MPRINT set

CALLED FROM

RRBRAK FOMCOM INOUT.

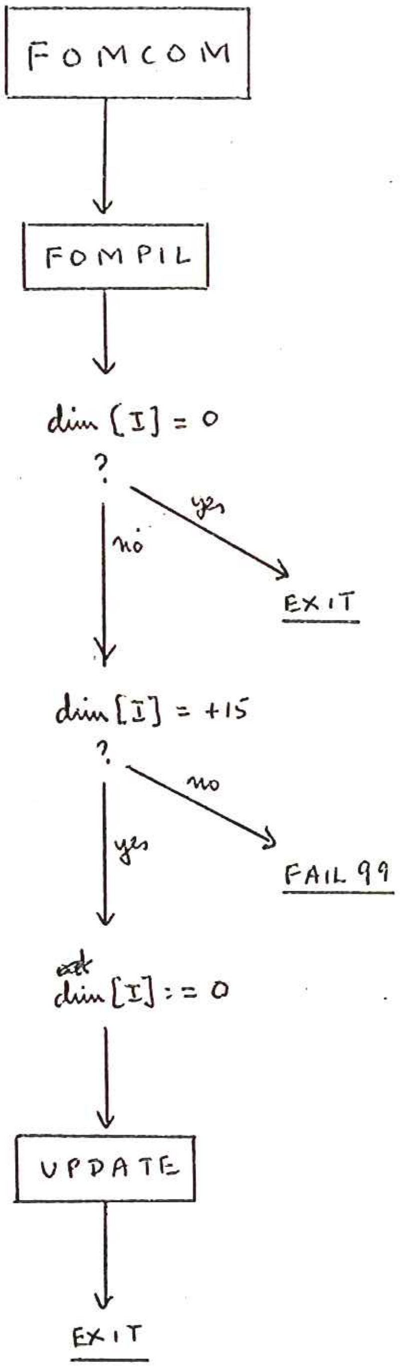
FOMPIL continued



this is suppressed inside PUNGRP in report mode

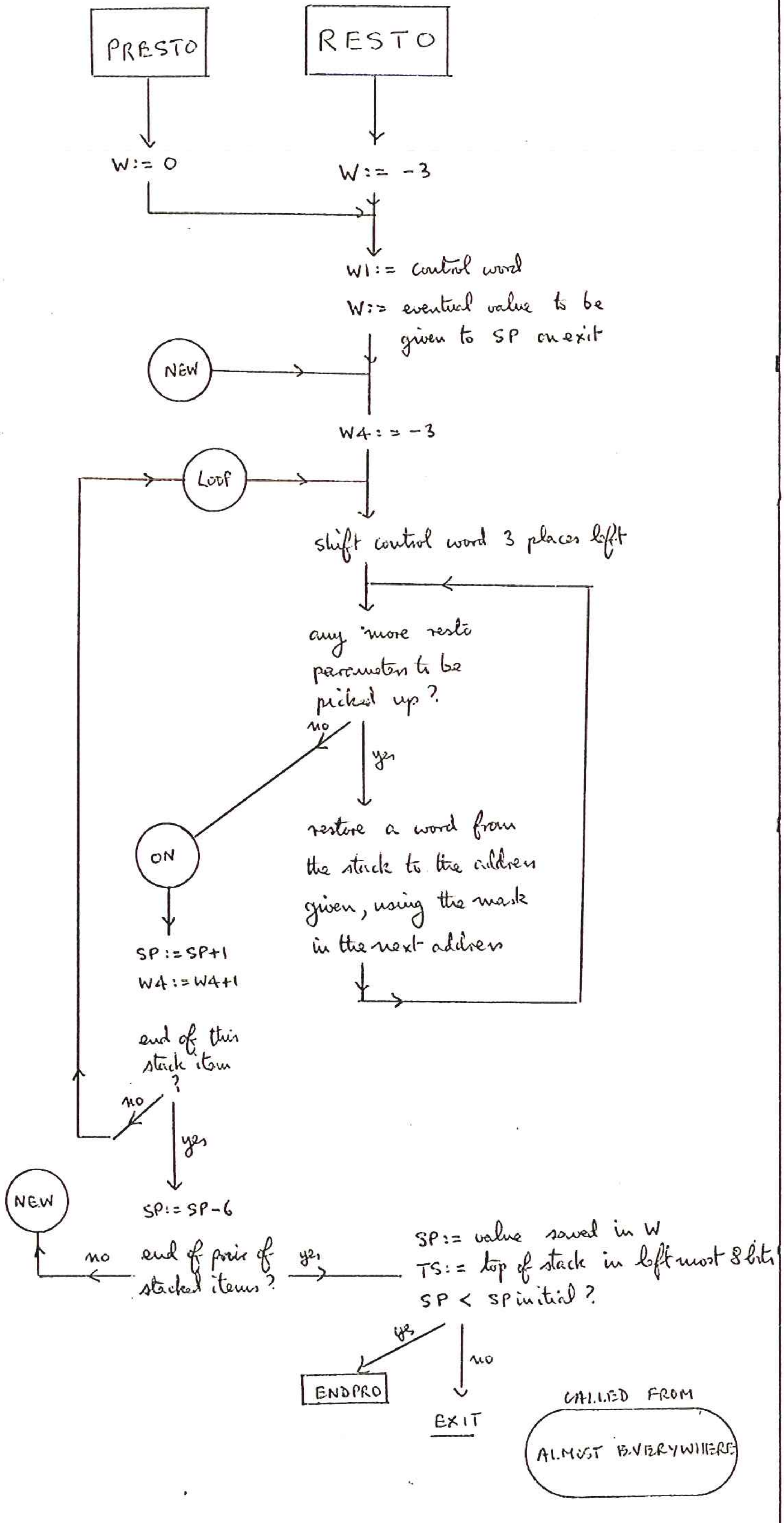
This is the normal library call which is dealt with at load time

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



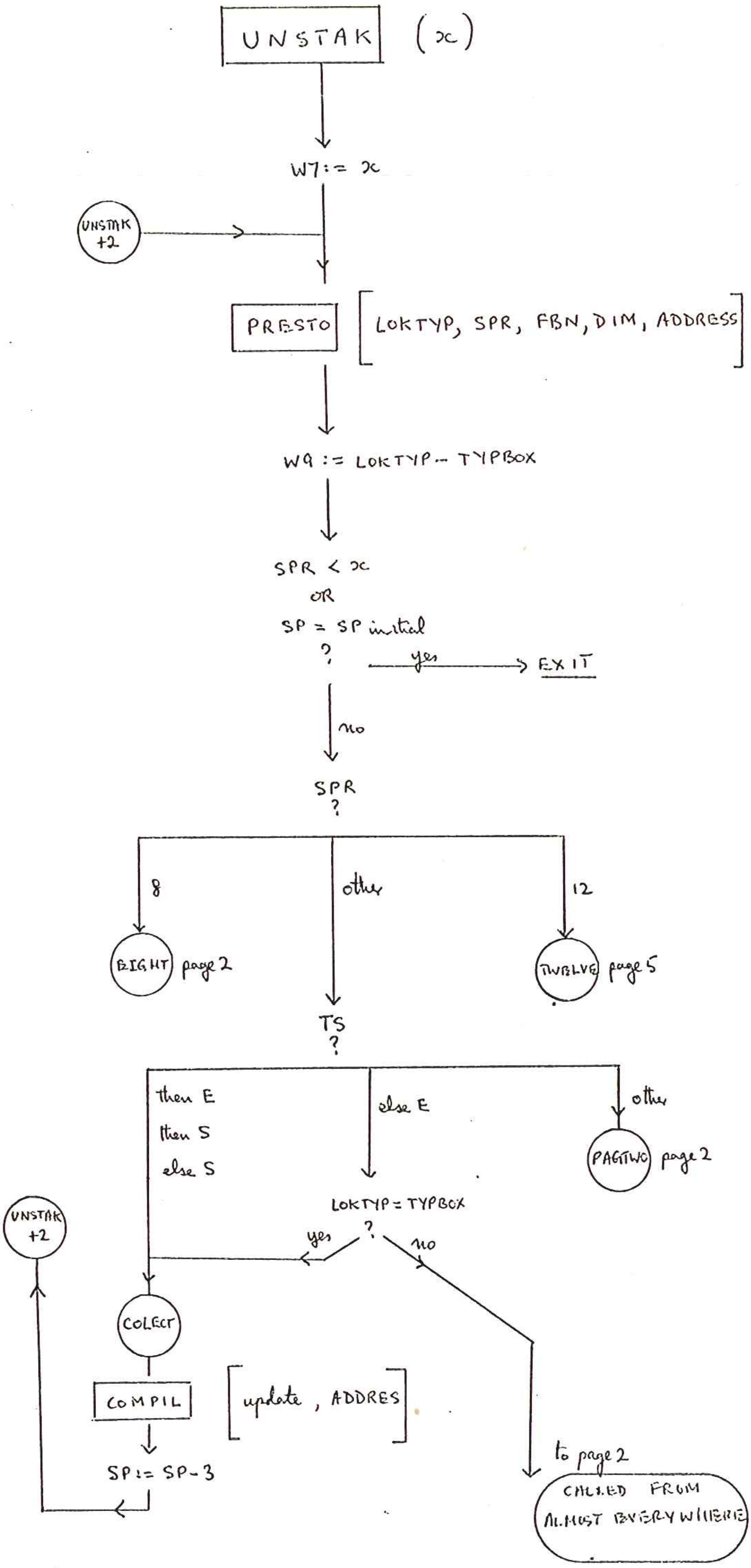
CALLED FROM

PRAMCH TAKID
ENDSTA


PRESTO
RESTO
W:= 0
W:= -3
W1:= control word
W:= eventual value to be given to SP on exit
NEW
W4:= -3
LOOP
shift control word 3 places left
any more resto parameters to be picked up?
no
yes
restore a word from the stack to the address given, using the mask in the next address
ON
SP:= SP+1
W4:= W4+1
end of this stack item?
no
yes
SP:= SP-6
NEW
no
end of pair of stacked items?
yes
SP:= value saved in W
TS:= top of stack in left most 8 bits
SP < SP initial?
yes
no
ENDPRO
EXIT
CALLED FROM
ALMOST EVERYWHERE

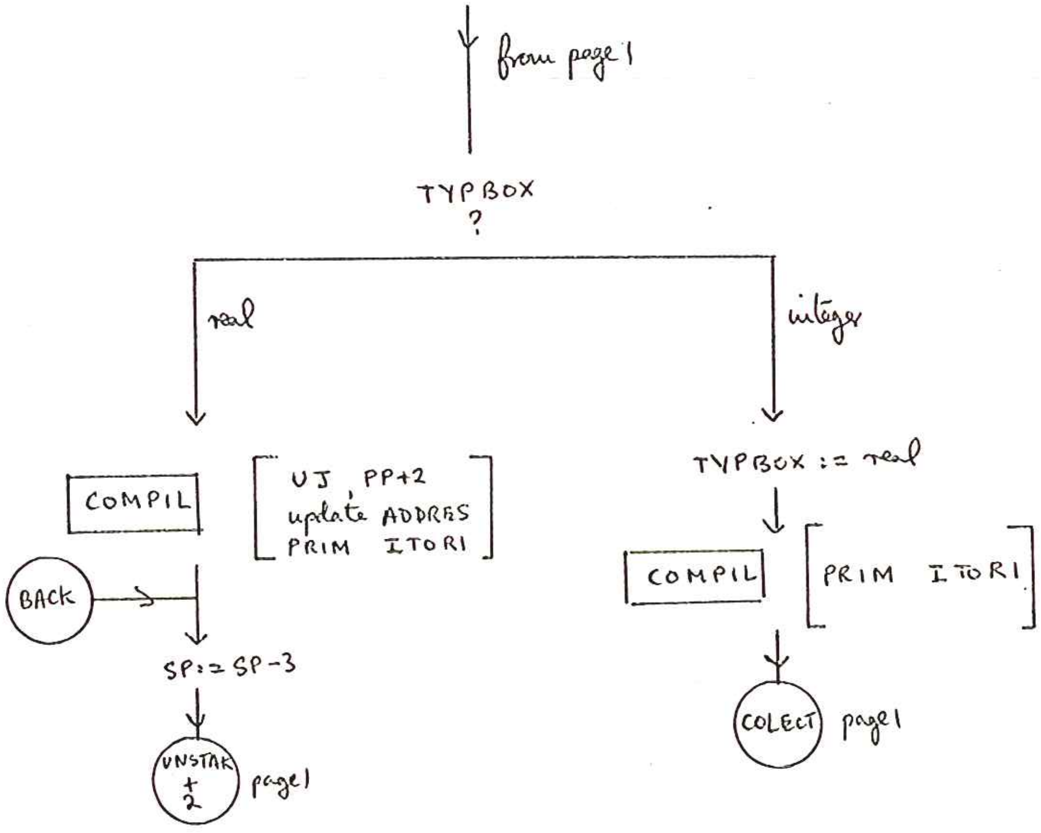
ENDPRO is reached to stop translation

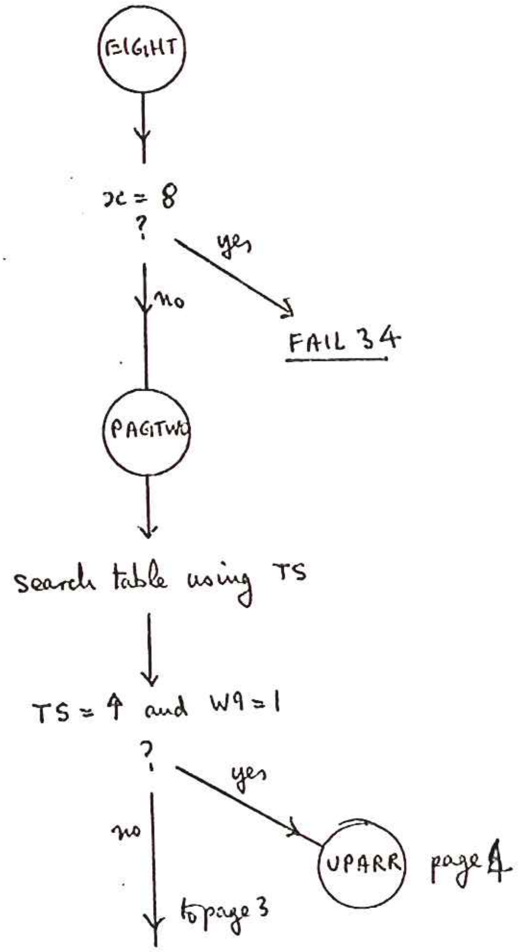
page 1 of 5



SPR is the stacked priority

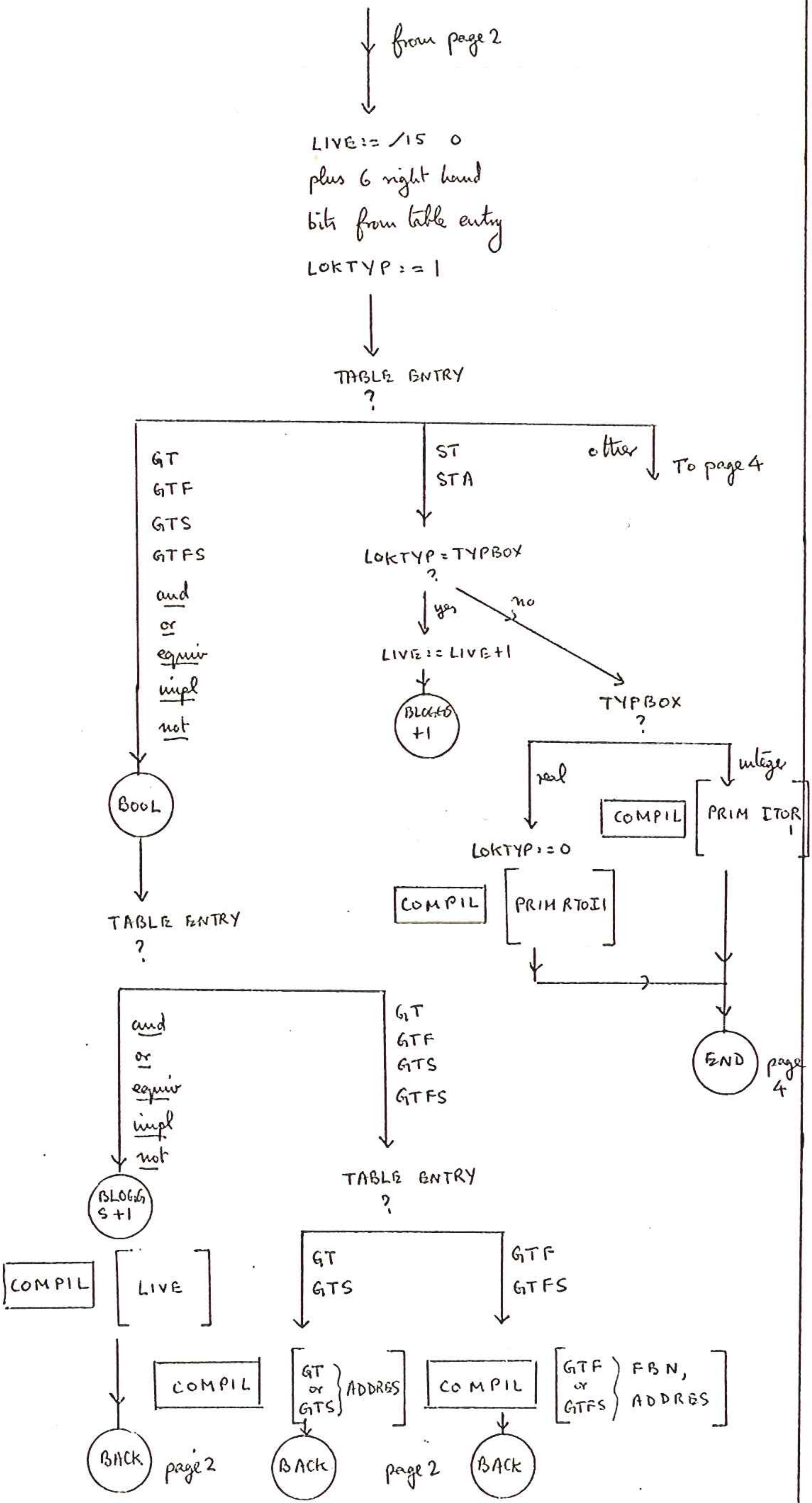
UNSTAK continued





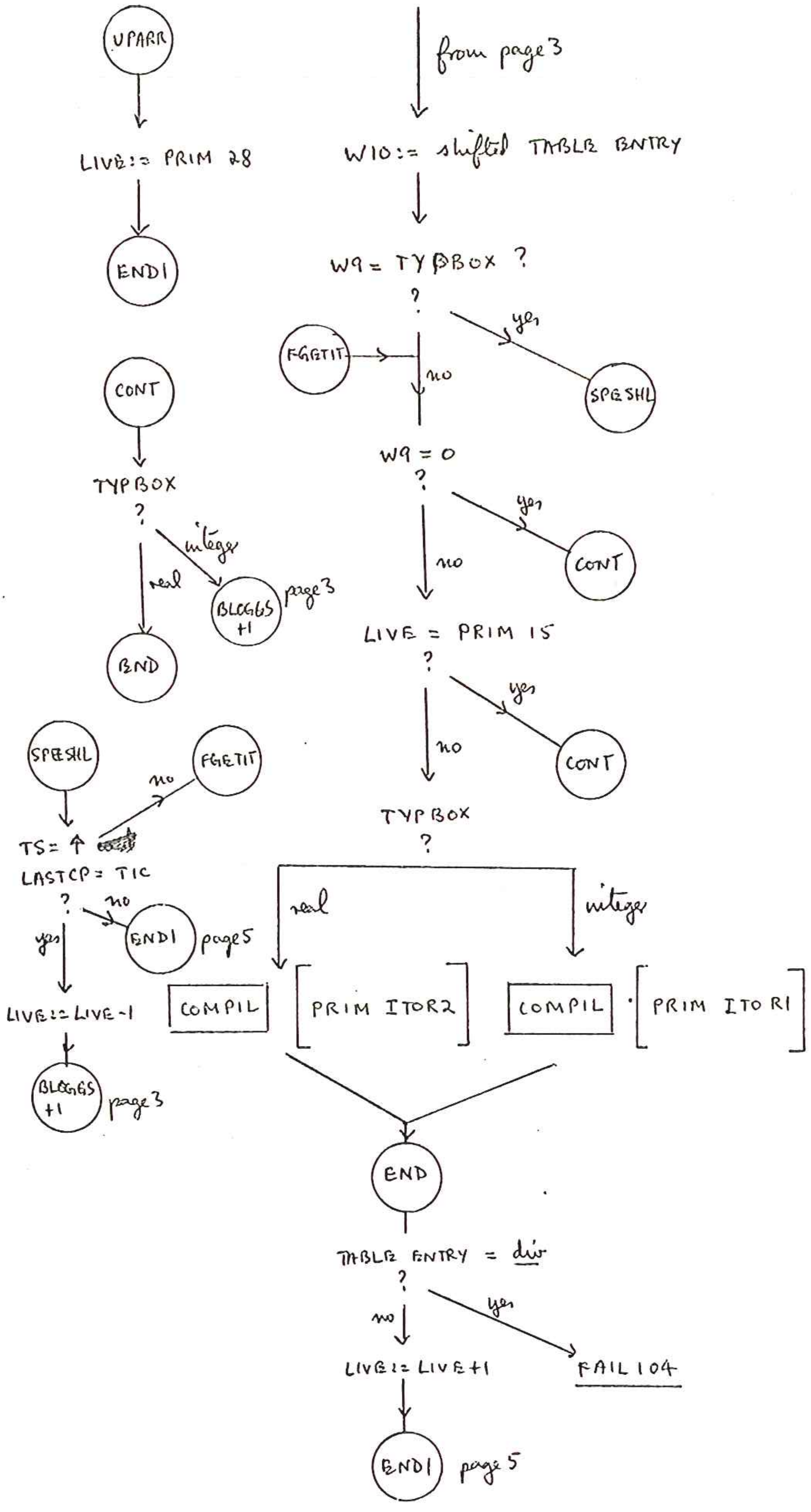
UNSTAK continued

page 3 of 5



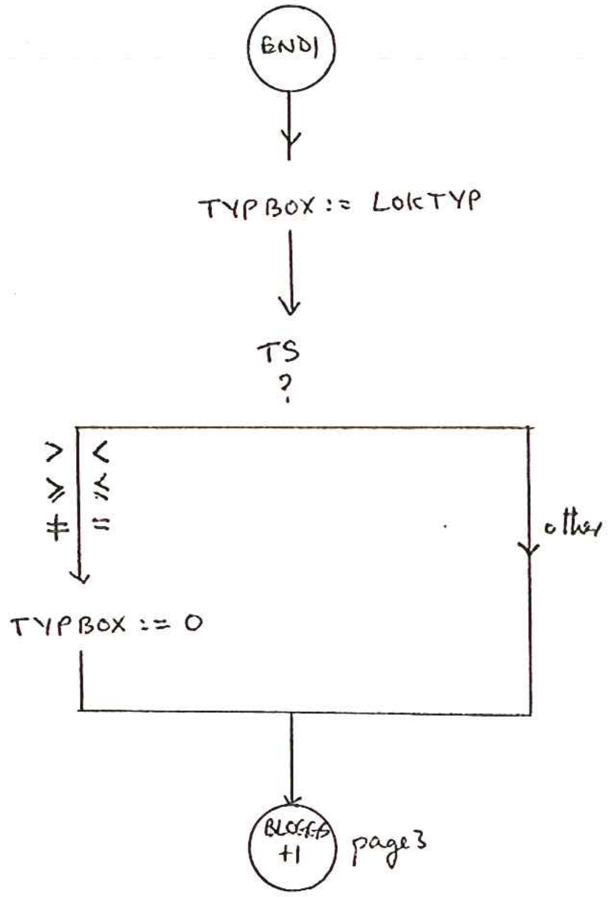
UNSTAK continued

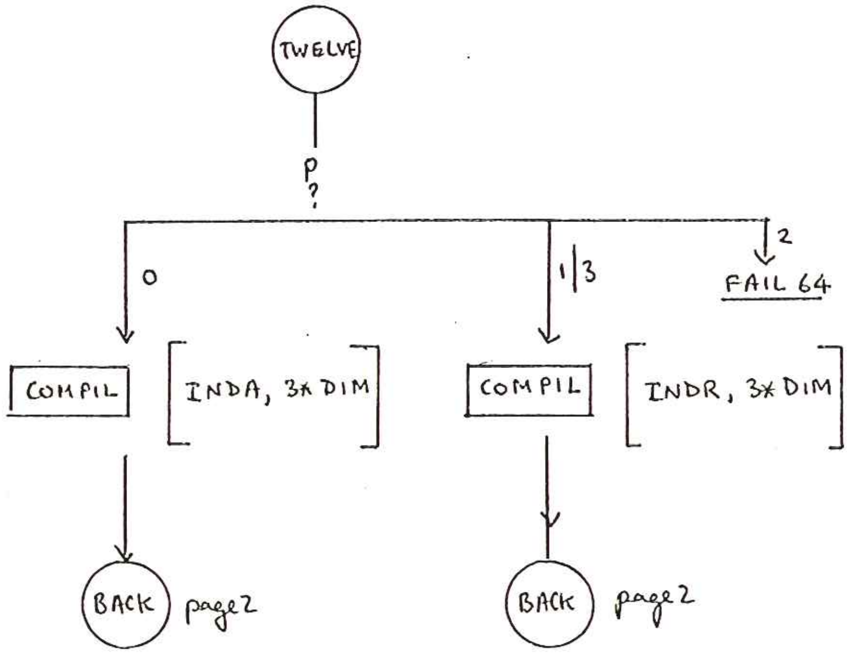
page 4 of 5

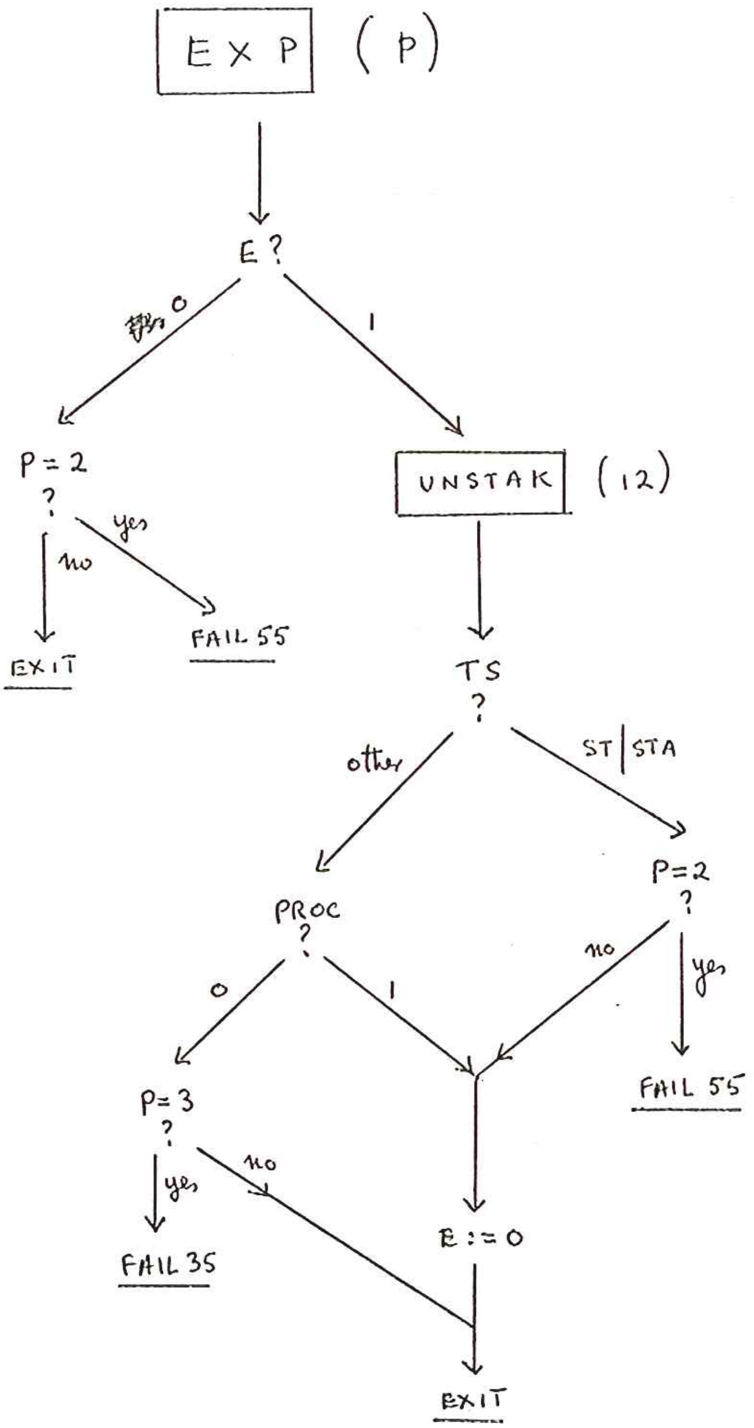


UNSTAK continued

page 5 of 5



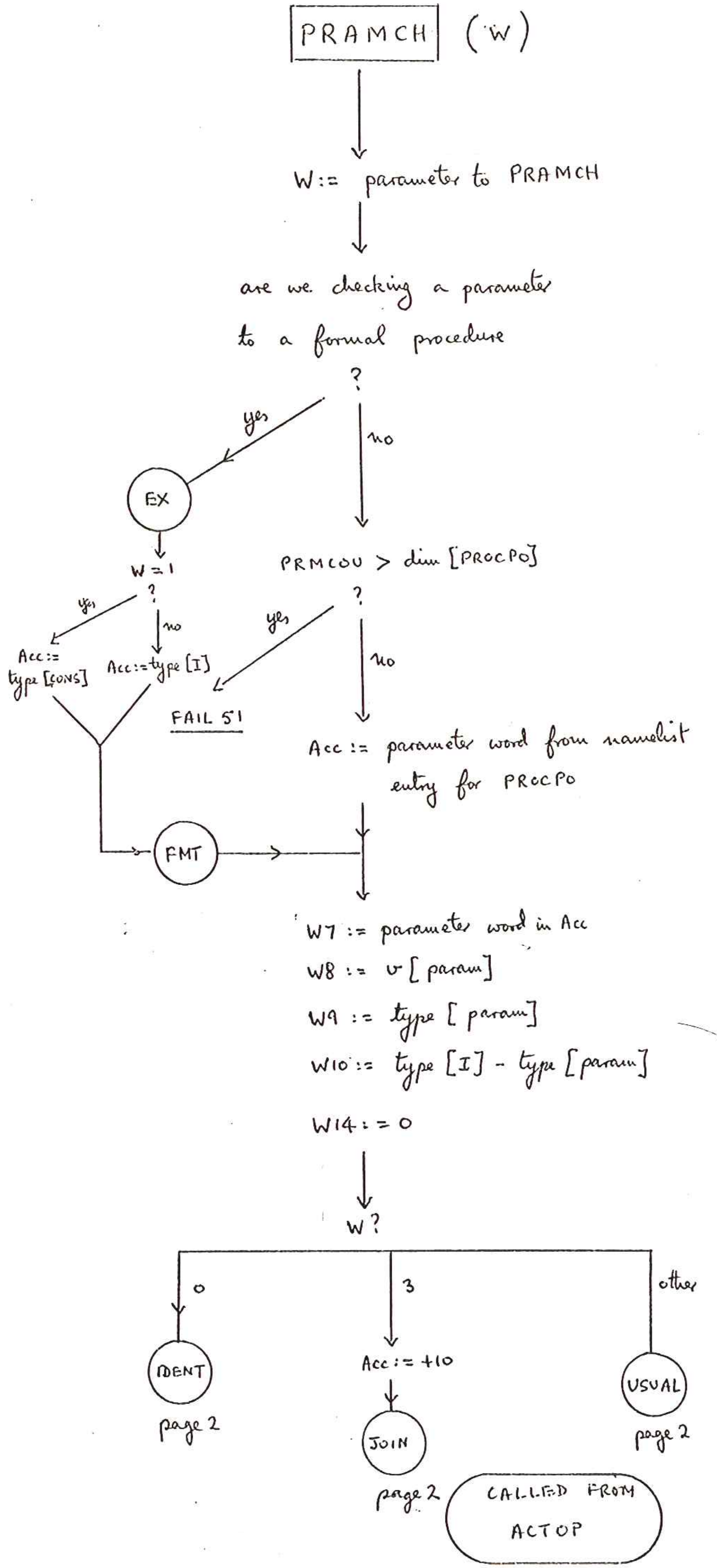




CALLED FROM

FOR GOTO IF AOP
RLT LOGOP COLON
LRBRAK ENDSTA

page 1 of 10



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

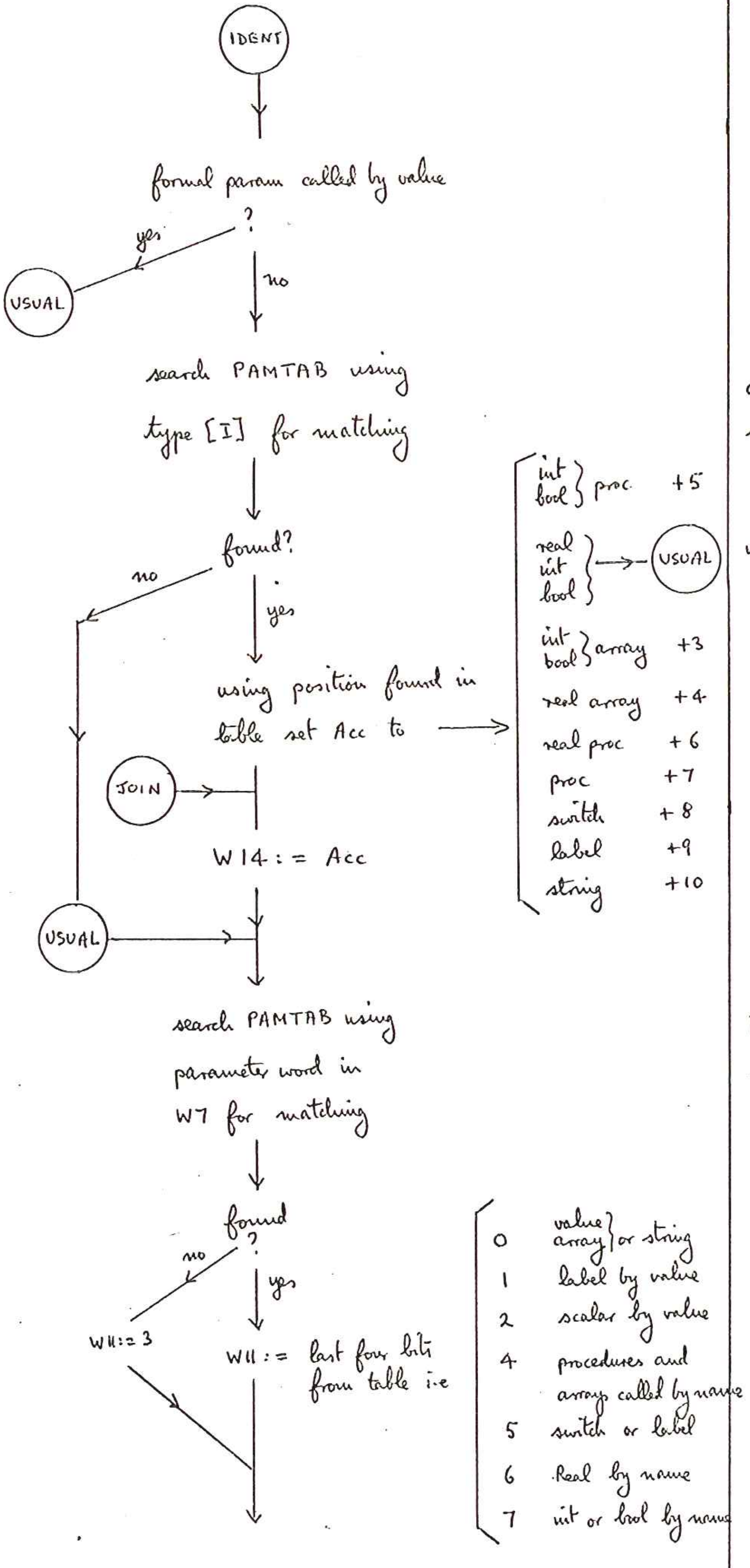
describes formal parameter

I points to actual param.

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

PRAMCH continued

page 2 of 10

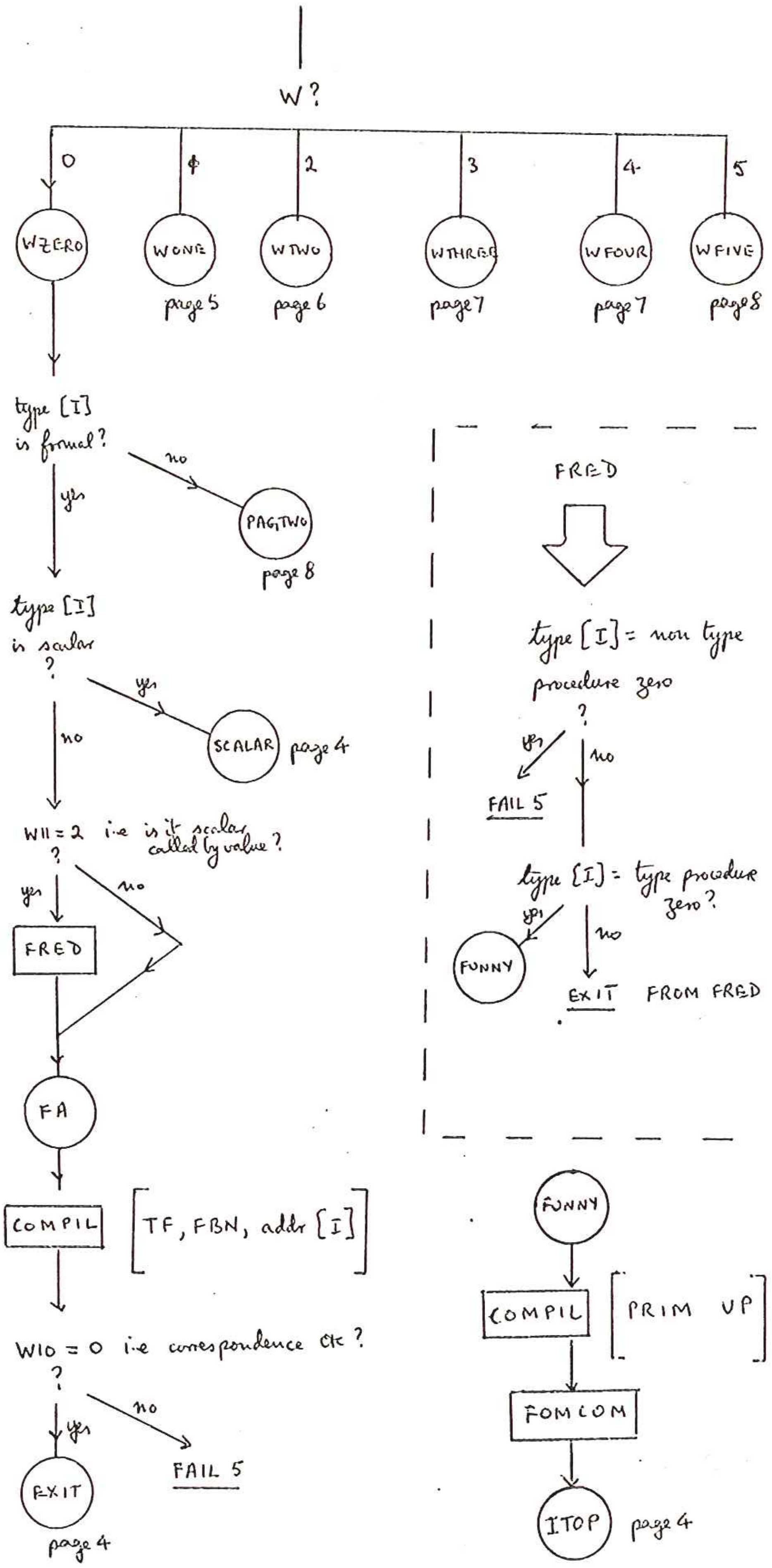


actual param search

W14 not altered in this case

formal param search

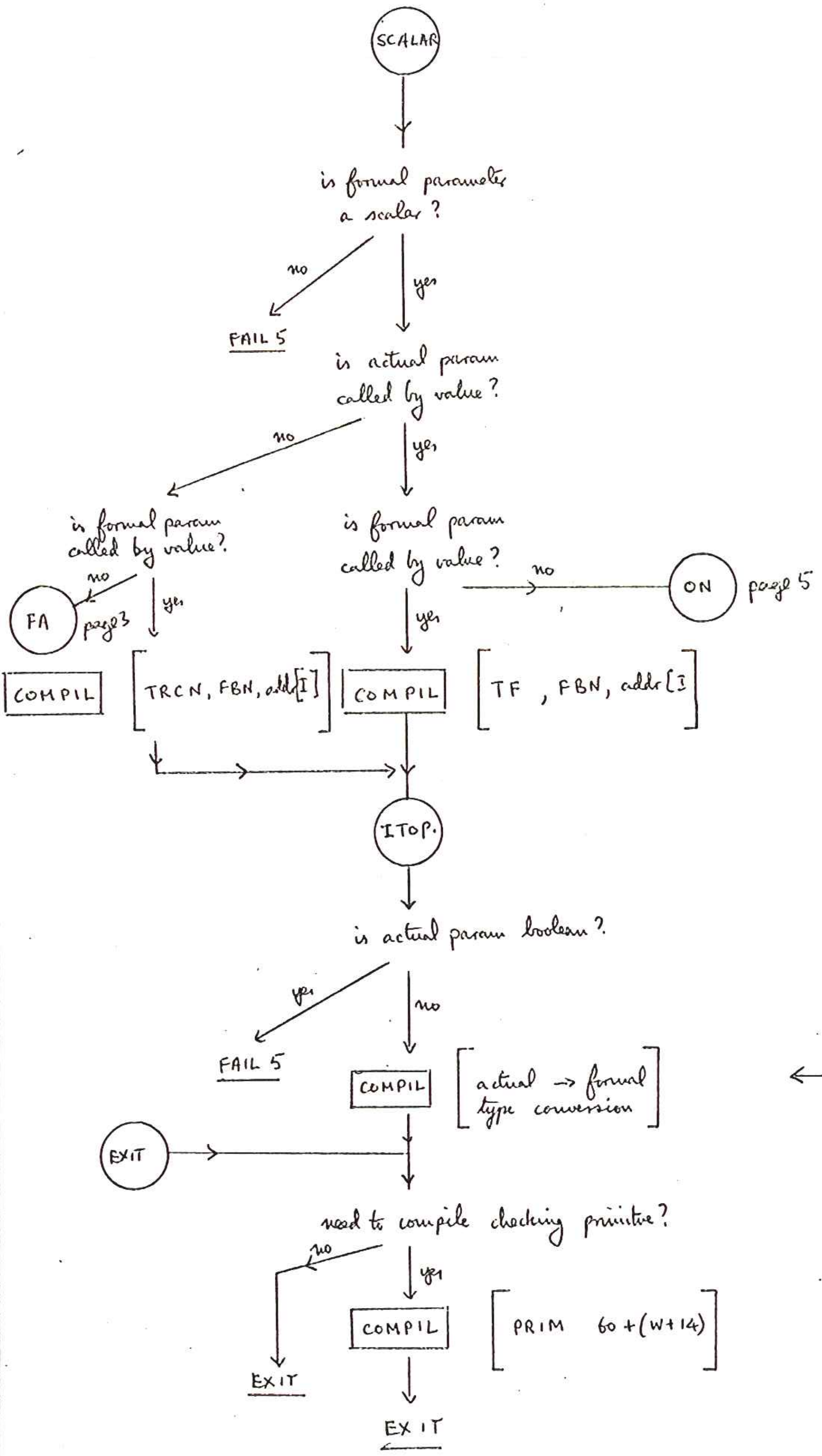
PRAMCH continued



local subroutine with exit to FUNNY

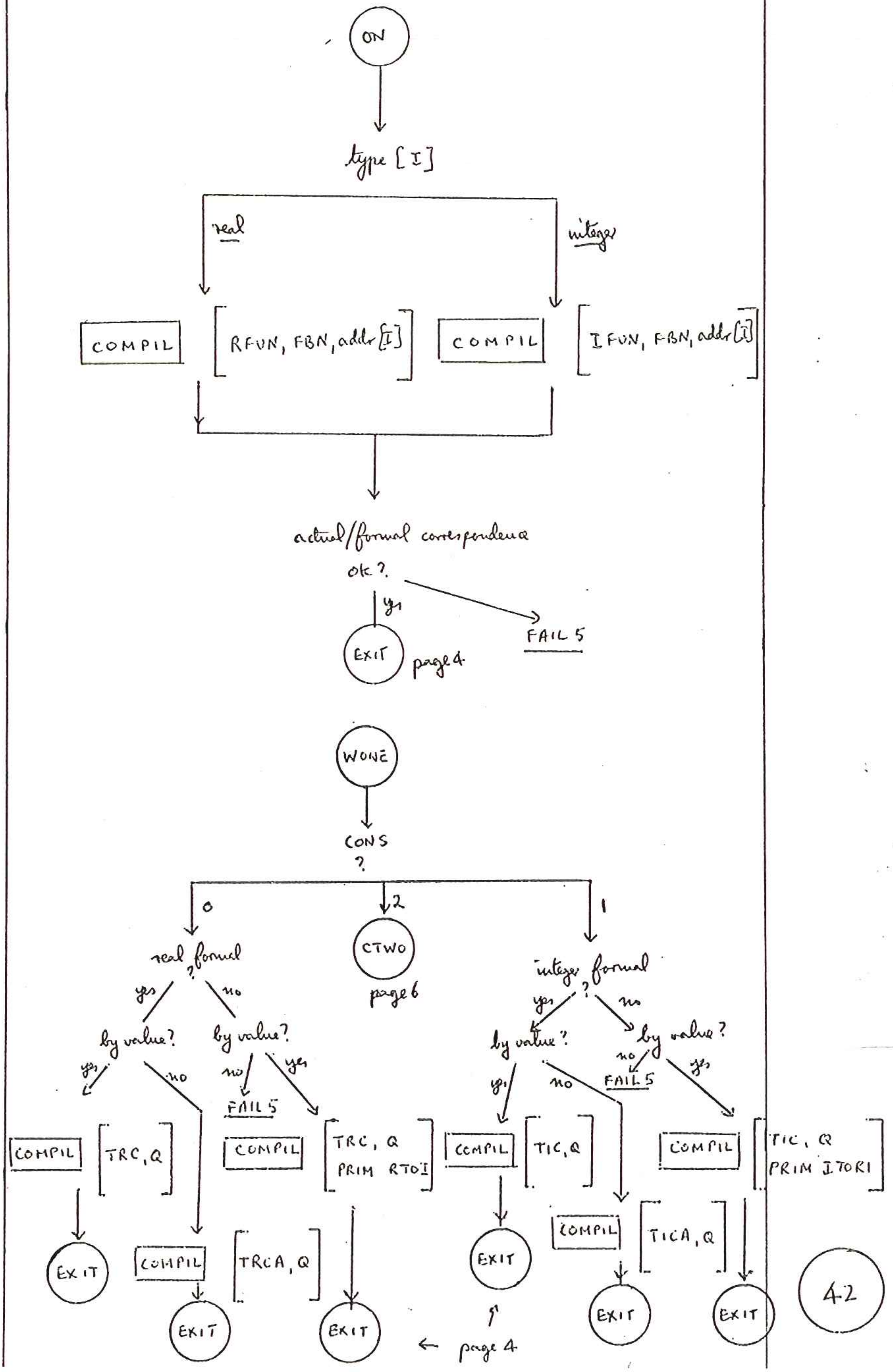
PRAMCH continued

a formal param is used here as an actual param

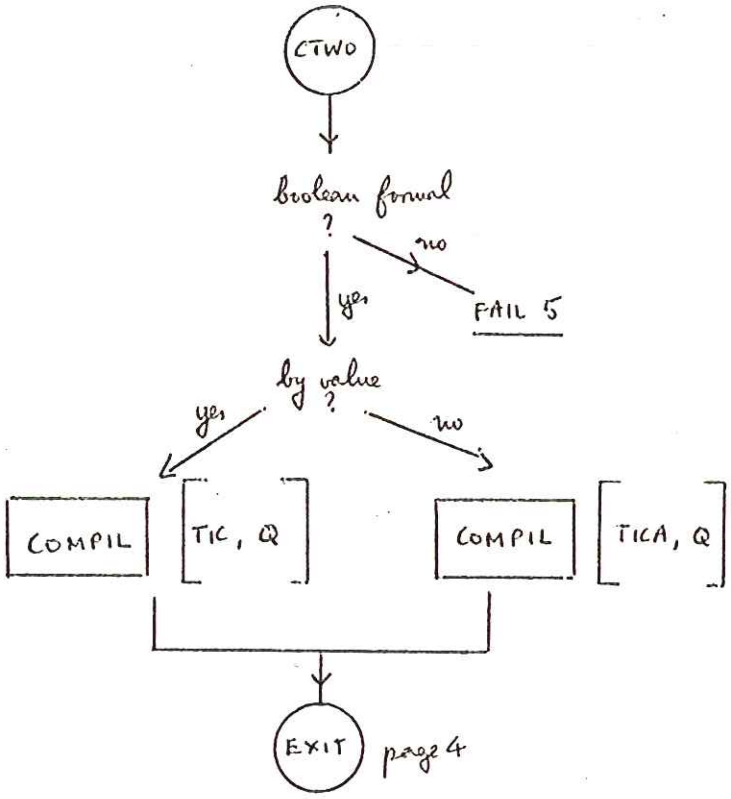

SCALAR
is formal parameter a scalar?
no
yes
FAIL 5
is actual param called by value?
no
yes
is formal param called by value?
no
yes
FA
page 3
COMPIL
TRCN, FBN, addr[I]
is formal param called by value?
no
yes
ON
page 5
COMPIL
TF , FBN, addr[I]
ITOP.
is actual param boolean?
yes
no
FAIL 5
COMPIL
actual → formal type conversion
EXIT
need to compile checking primitive?
no
yes
COMPIL
PRIM 60 + (W + 14)
EXIT
EXIT

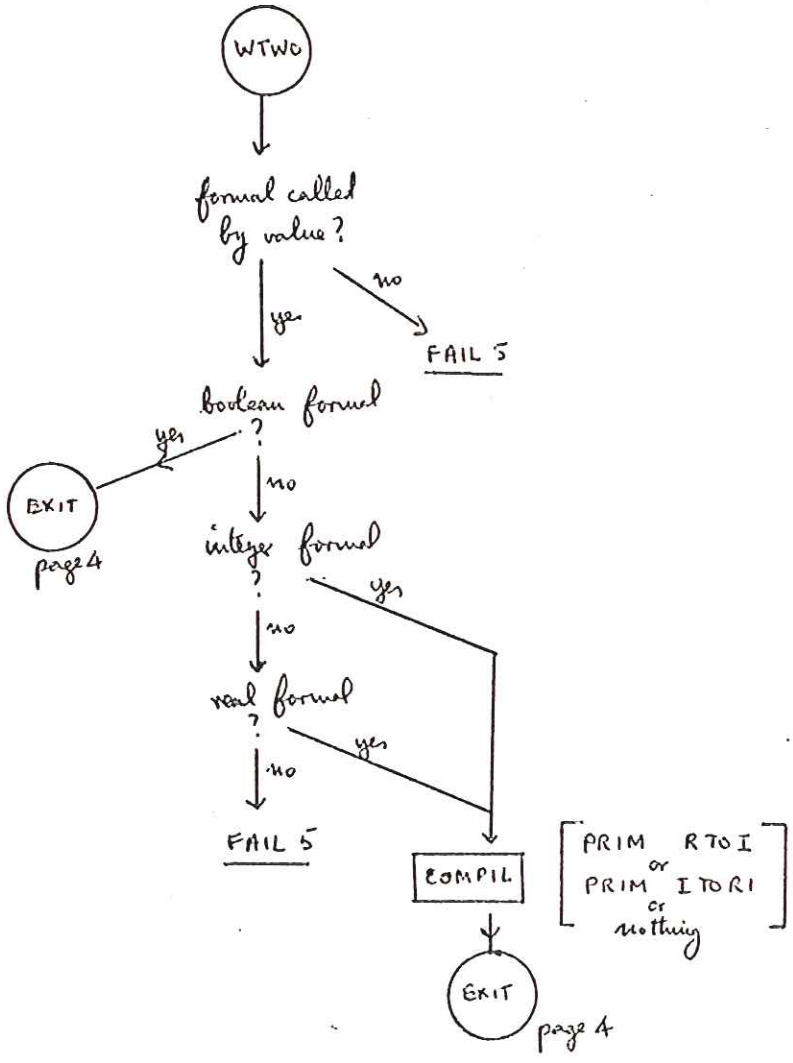
if no conversion is necessary this is skipped

PRAMCH continued



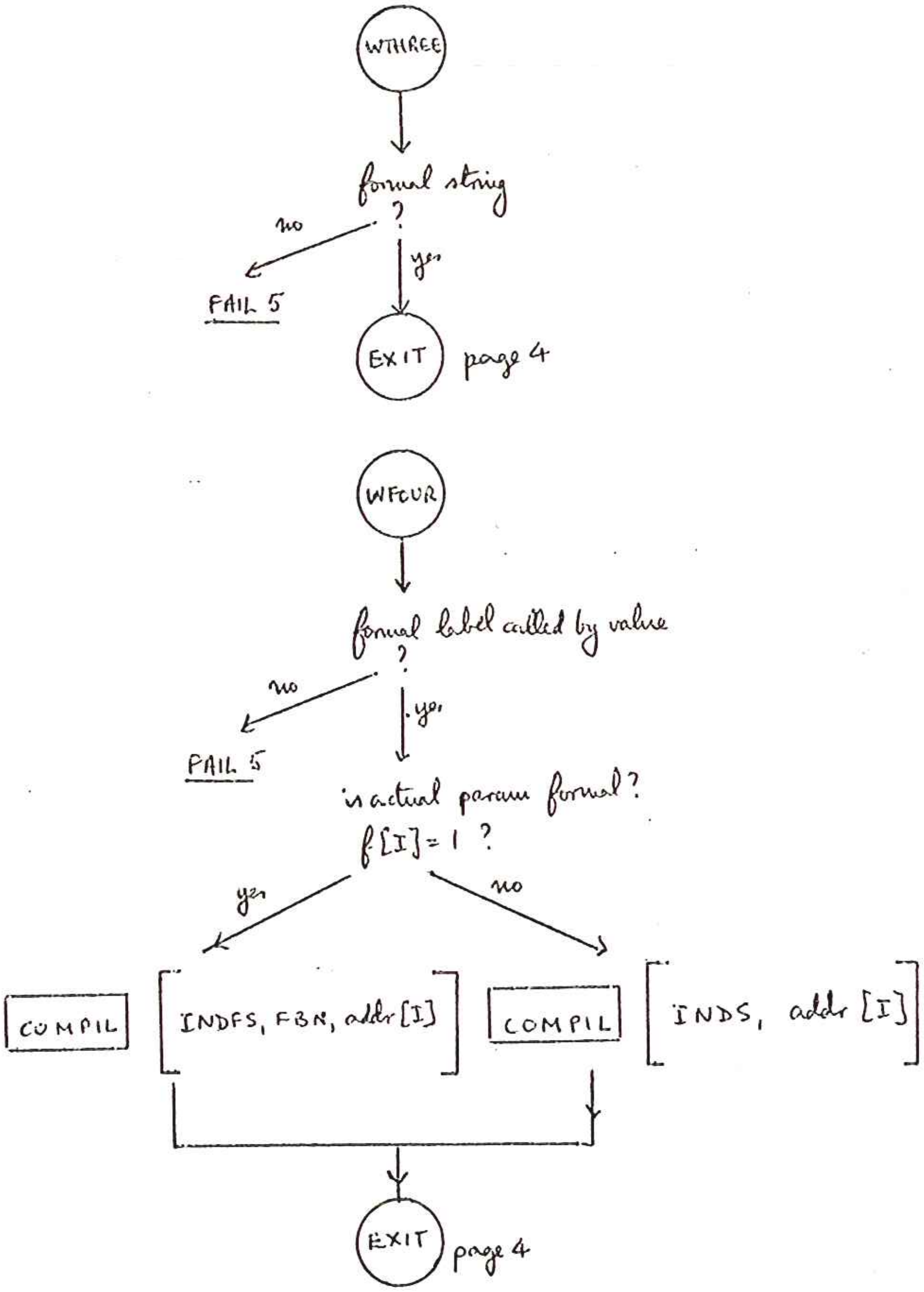
PRAMCH continued



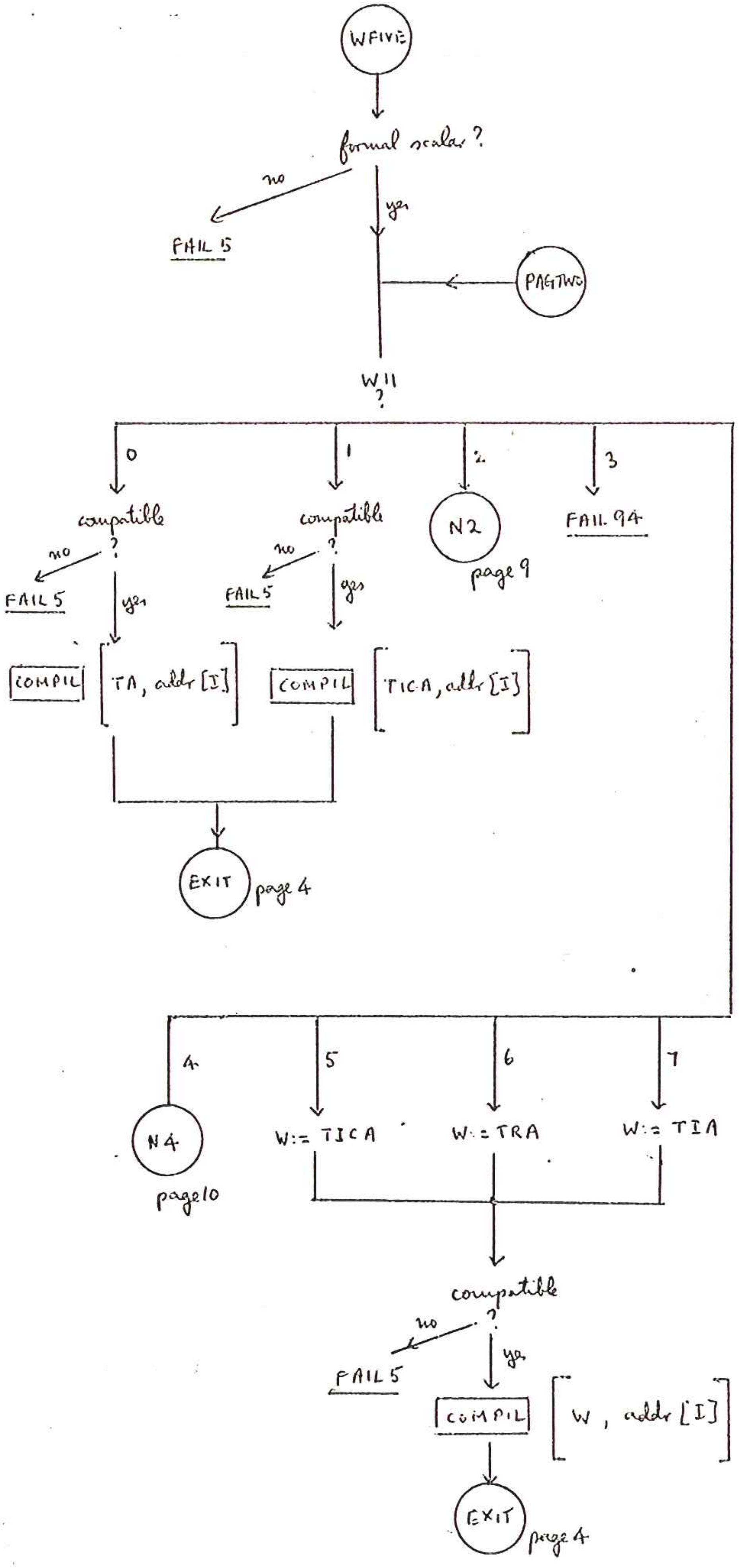


depends on TYPBOX to convert type from actual to formal

PRAMCH continued

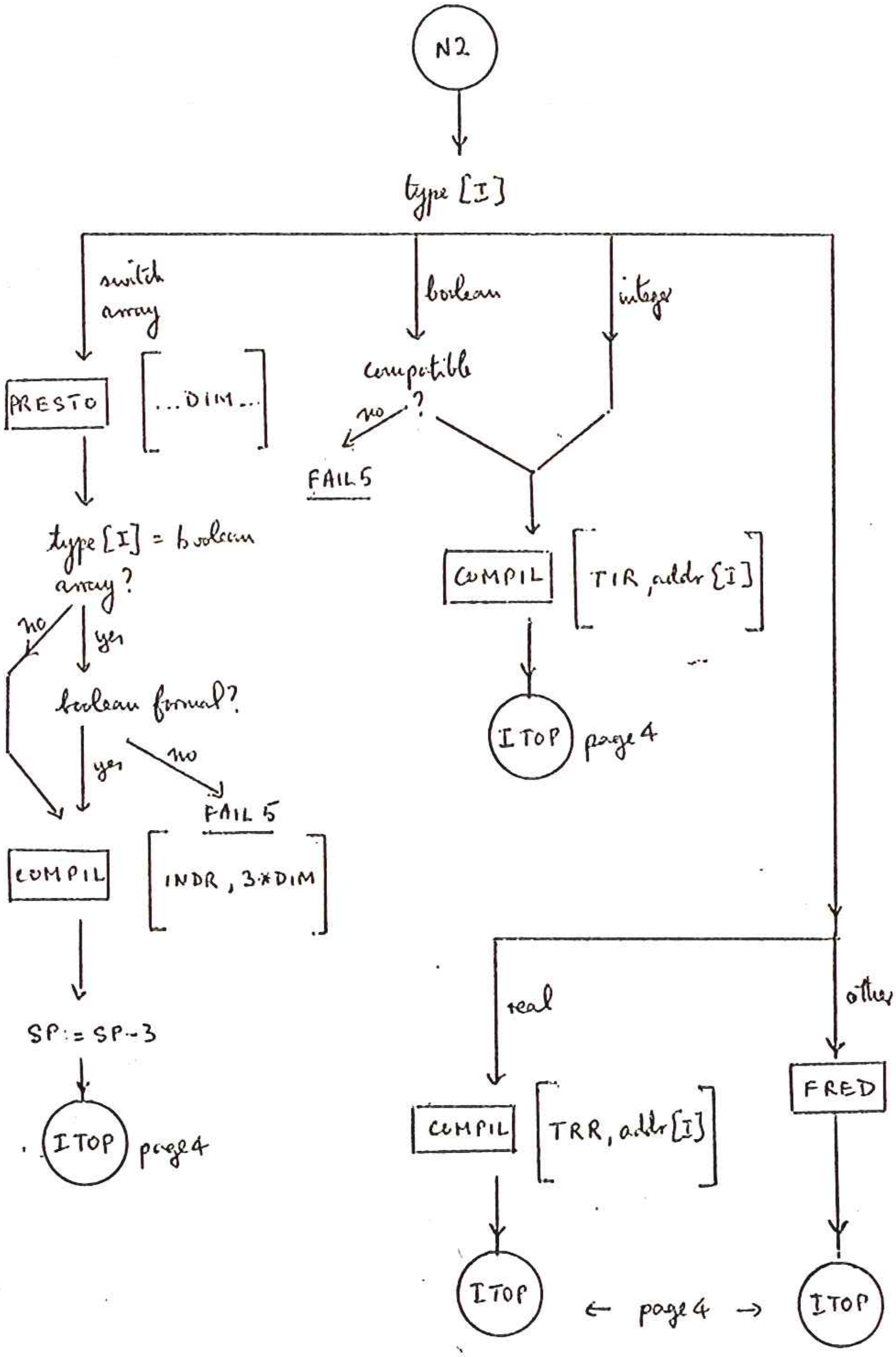


PRAMCH continued

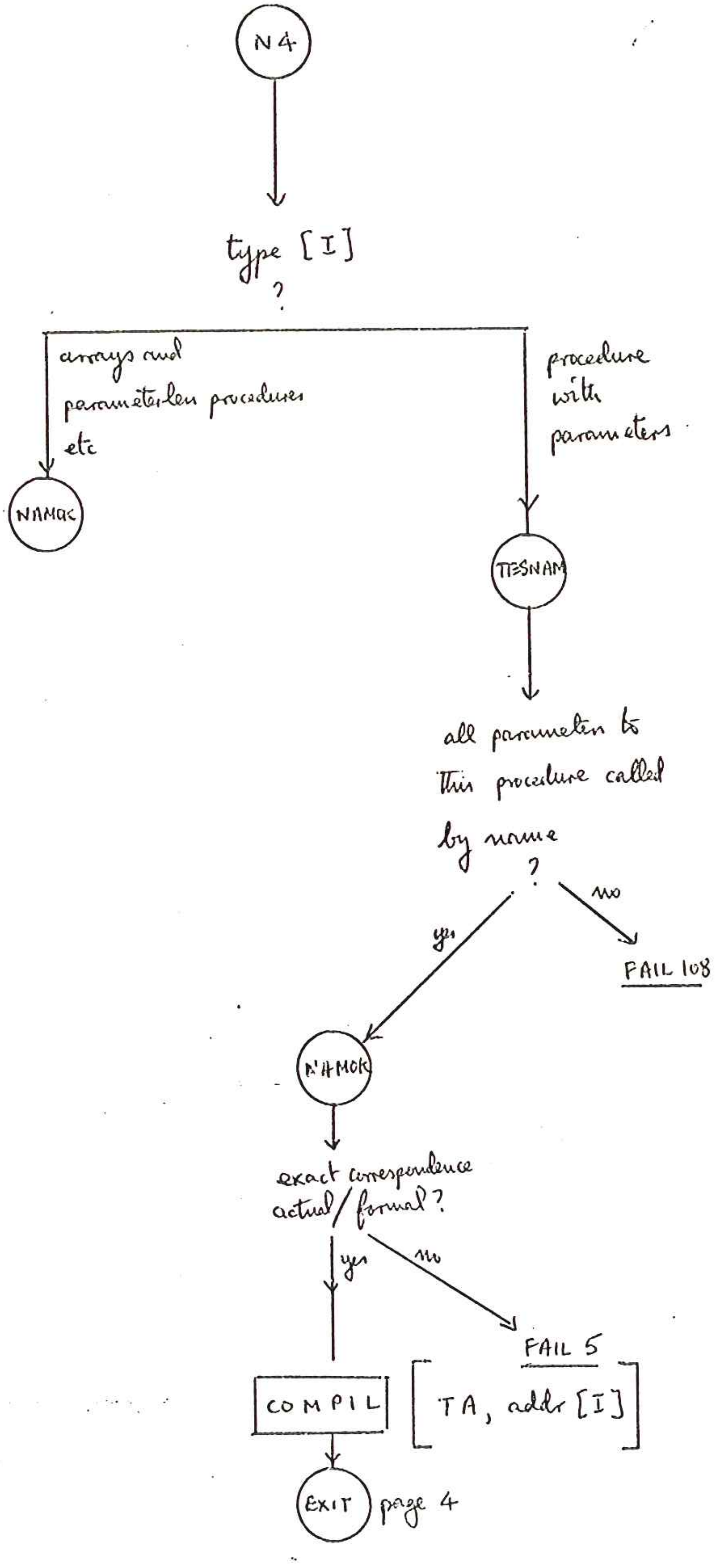


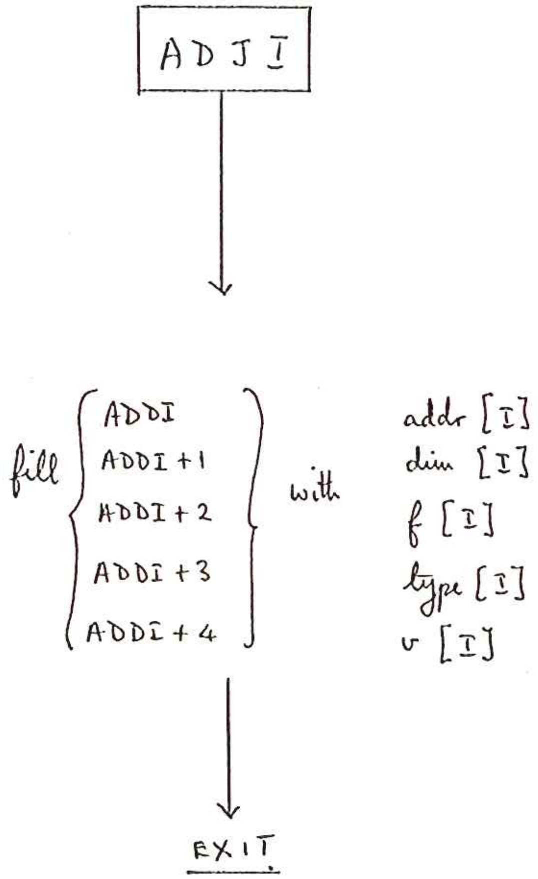
PRAMCH continued

page 9 of 10



PRAMCH continued





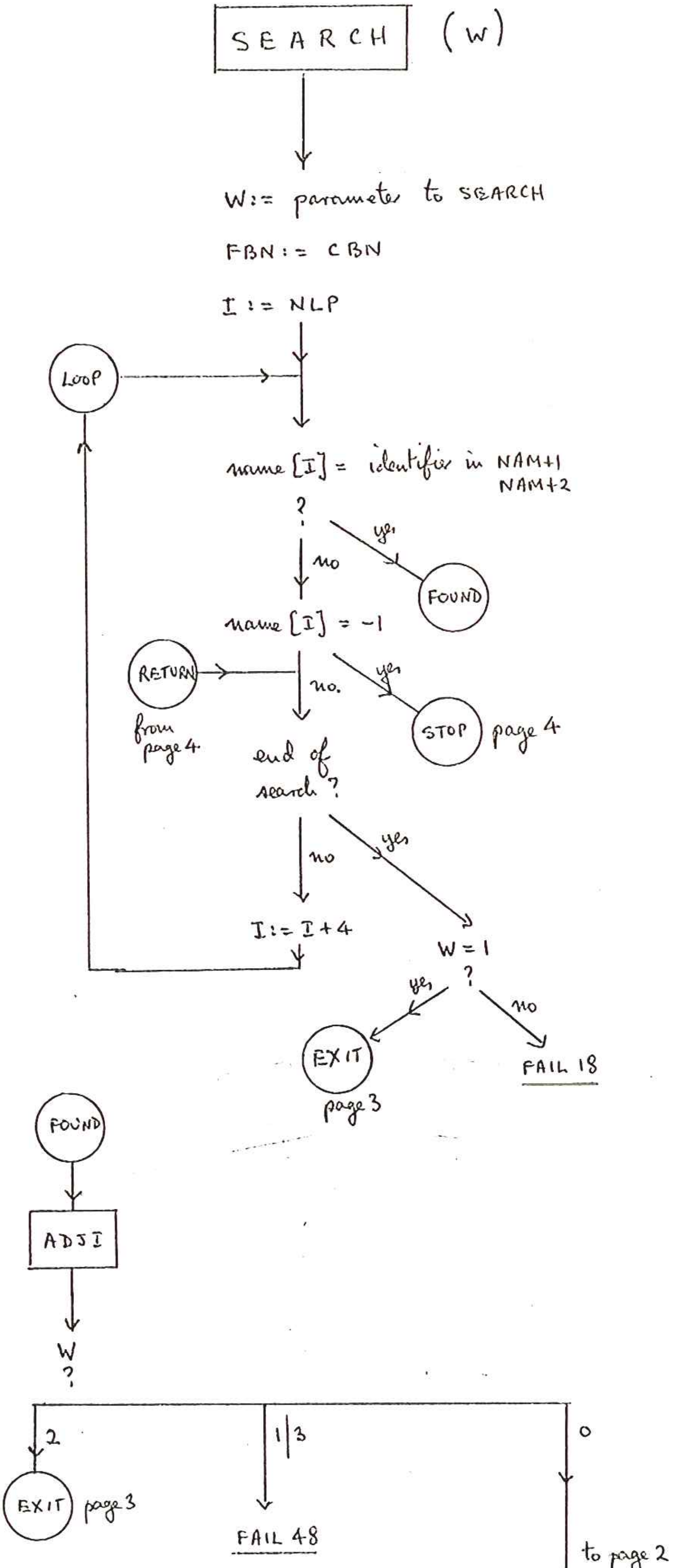
f = 1 if formal

v = 1 if called by value

CALLED FROM

RRBRAK
SEARCH

page 1 of 4



-1 represents stopper

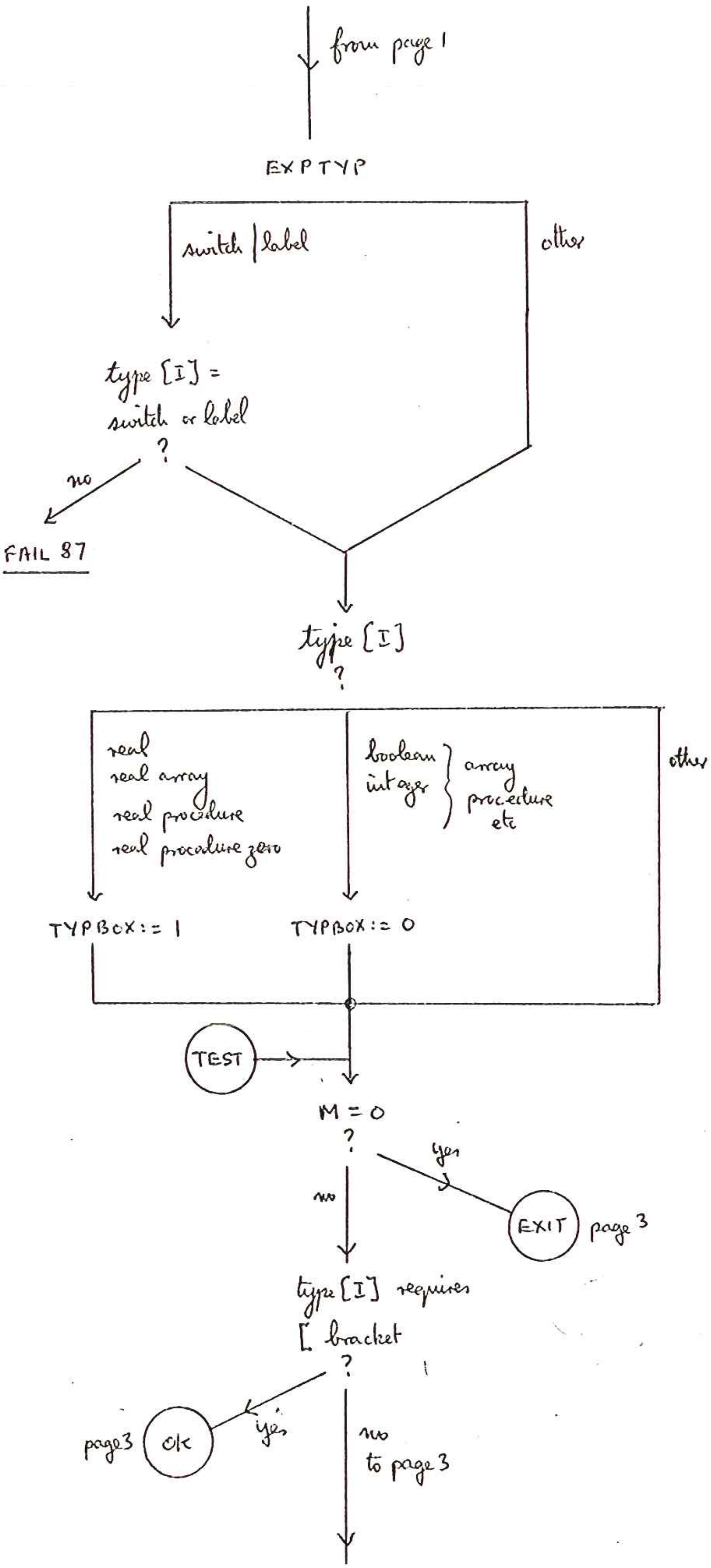
search ends at 7995

CALLED FROM

PROCED COLON TAKID
LRBRAK ACTOP
DECL INCUT

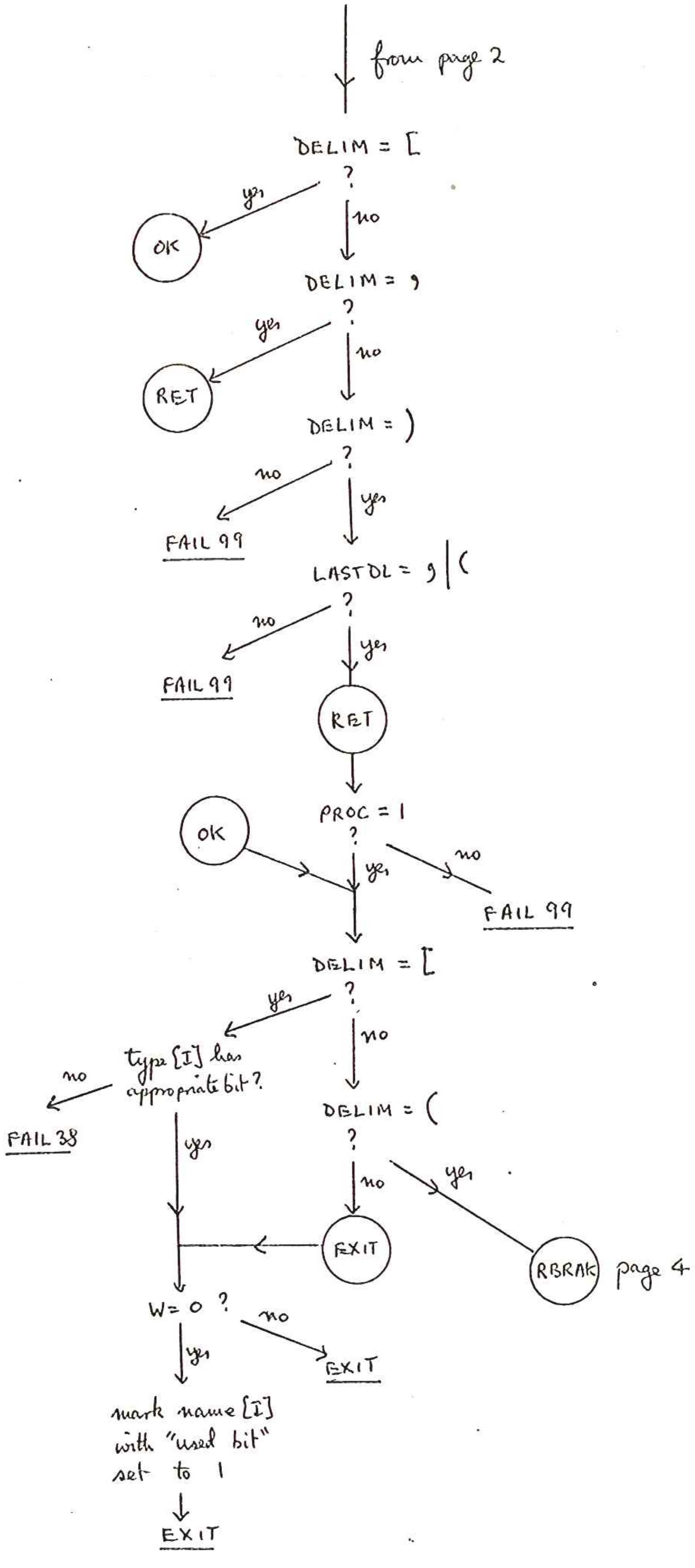
SEARCH continued

page 2. of 4.

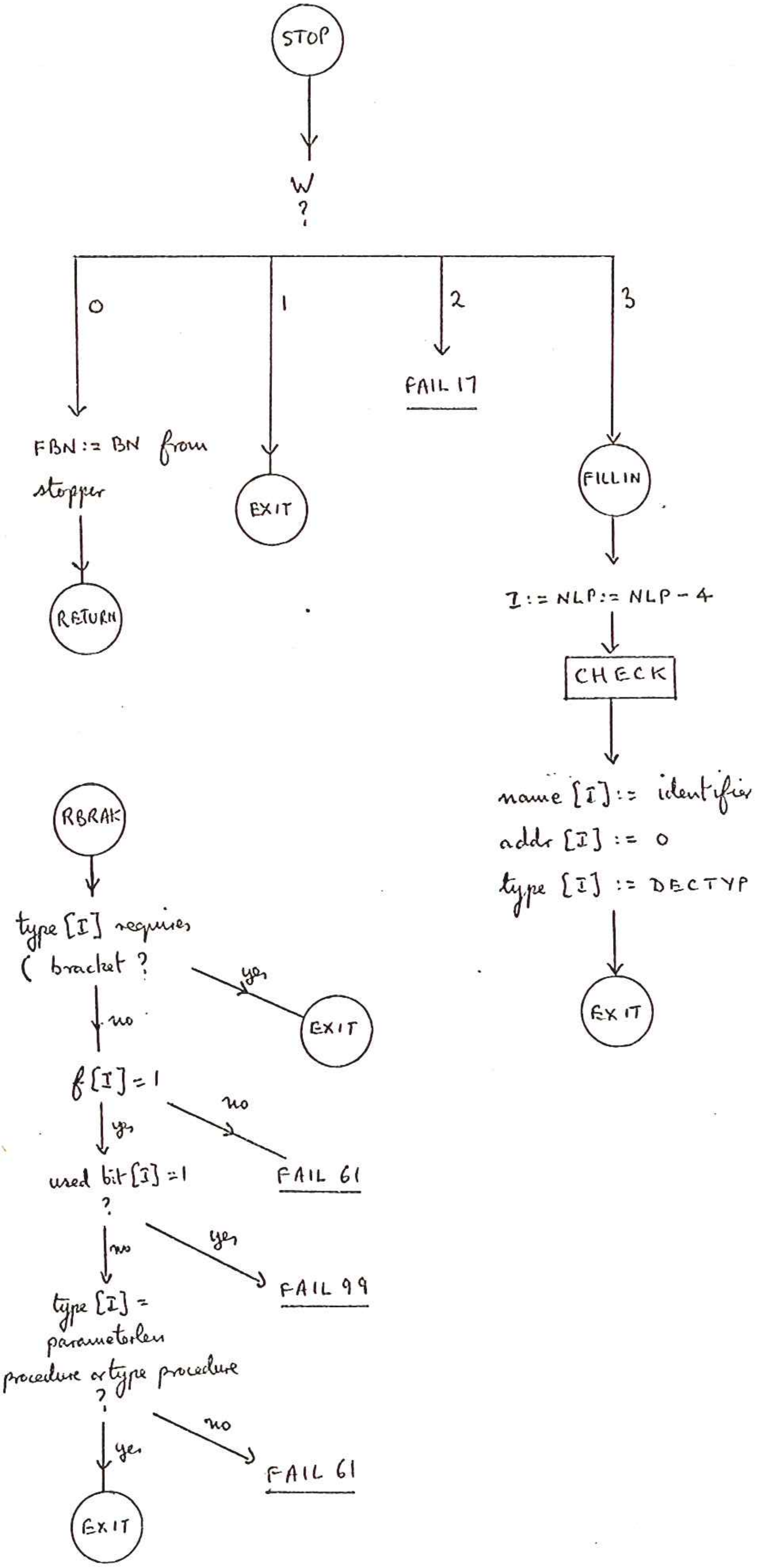


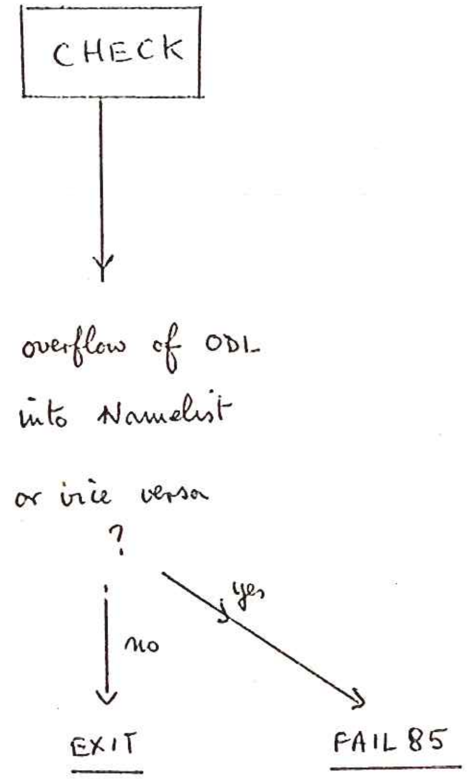
SEARCH continued

page 3 of 4



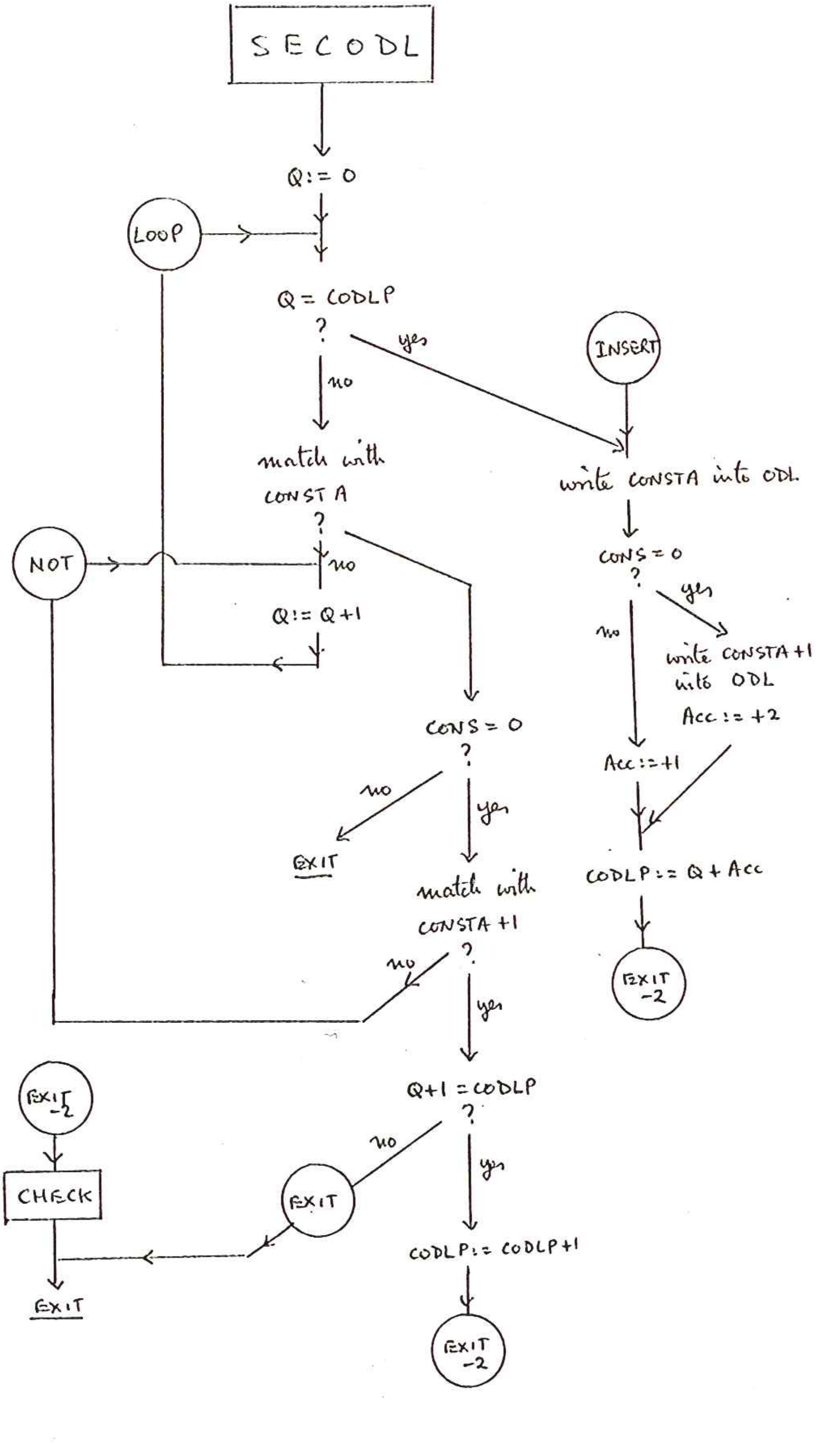
SEARCH continued





CALLED FROM

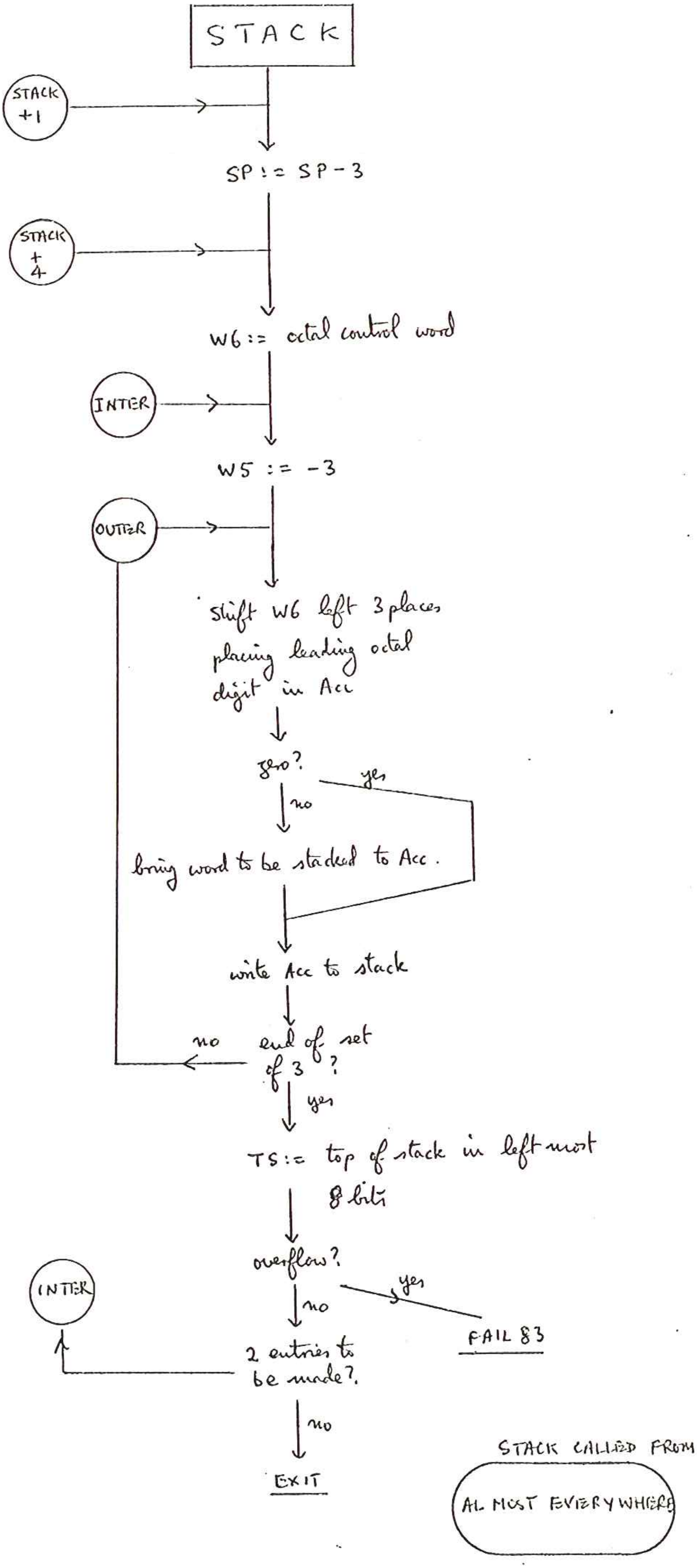
FOR SEARCH
SECODL DEC



real constant
in CONSTA
and
CONSTA +1

CALLED FROM

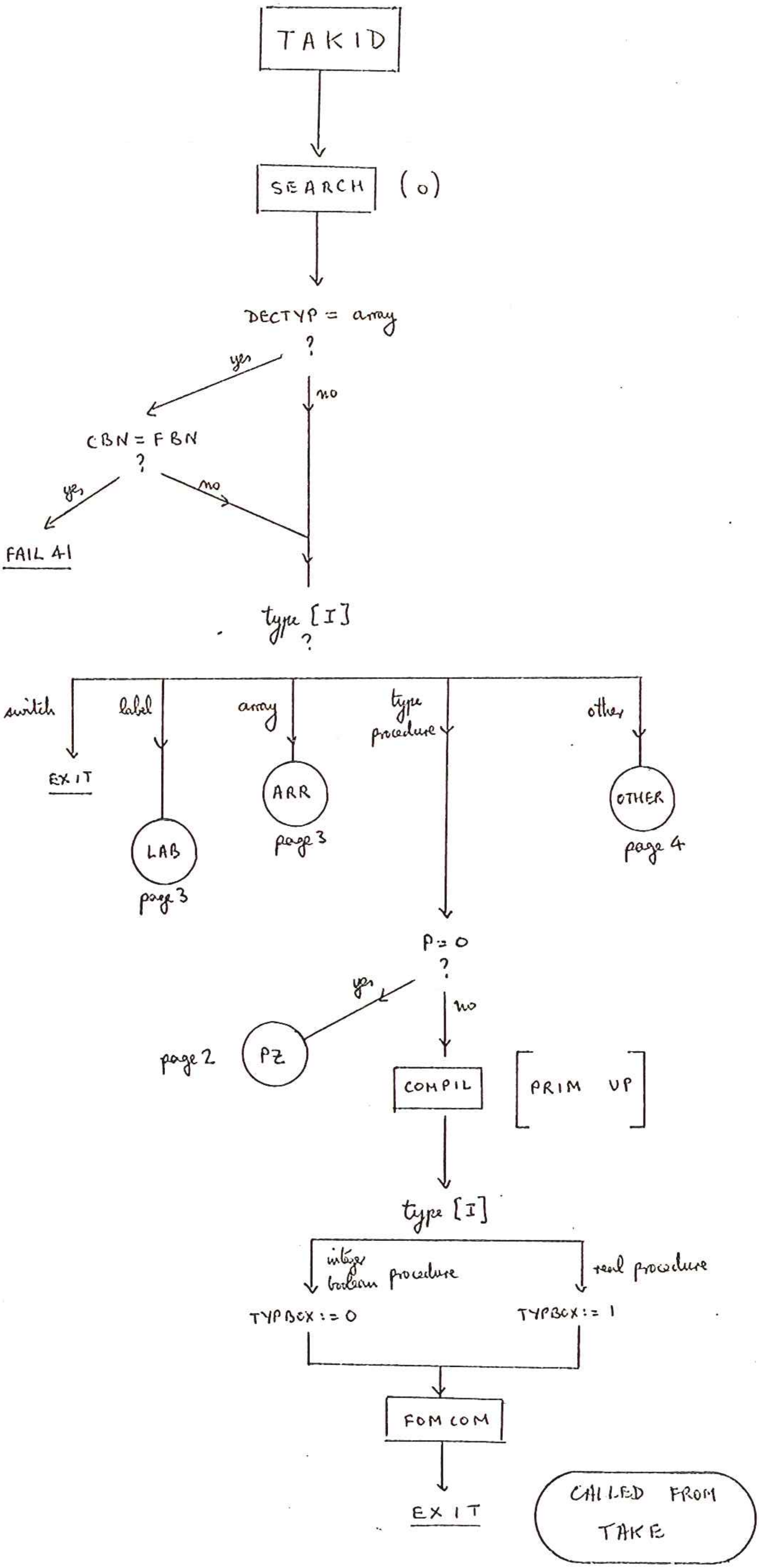
TAKE ACTOP



Overwrites top of stack with new entry

Normal Entry

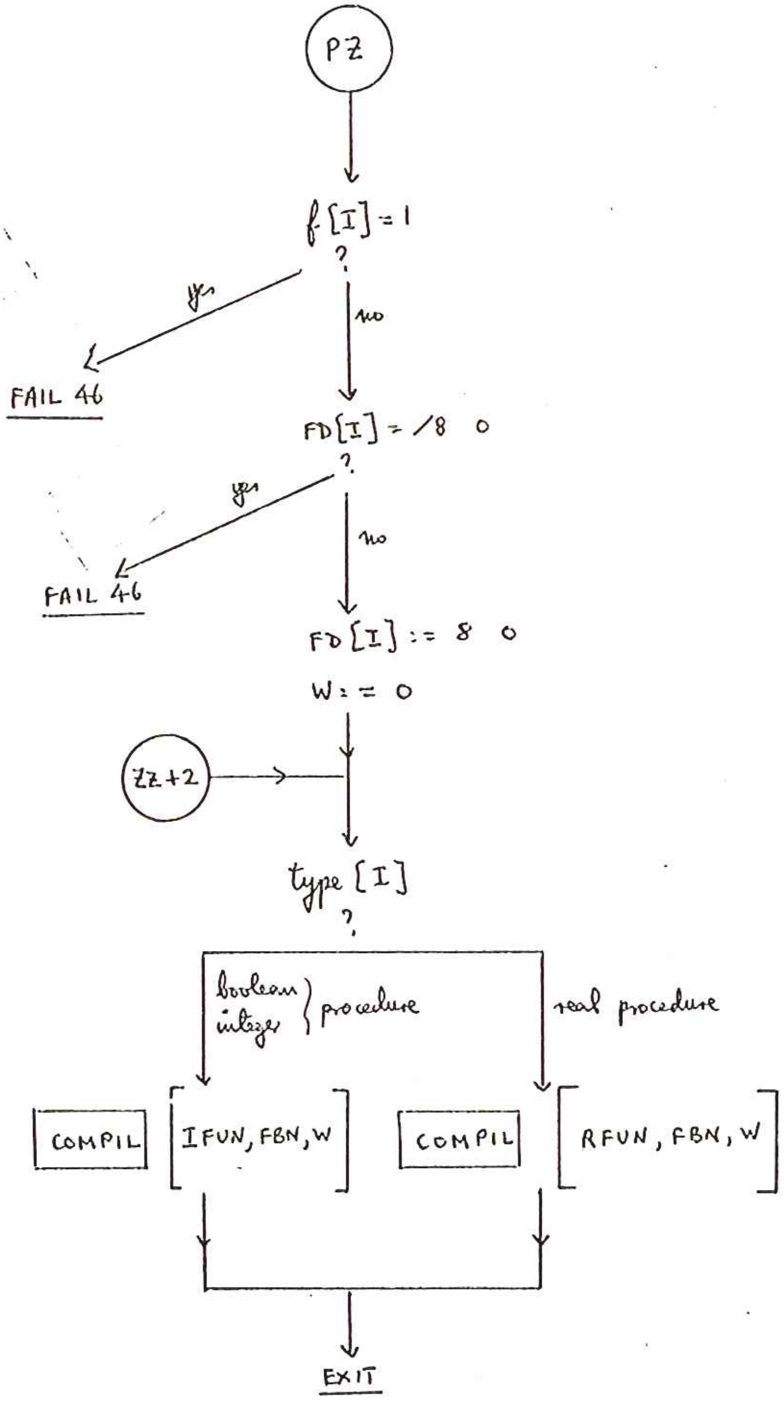
page 1 of 5



TAKID
SEARCH
(o)
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

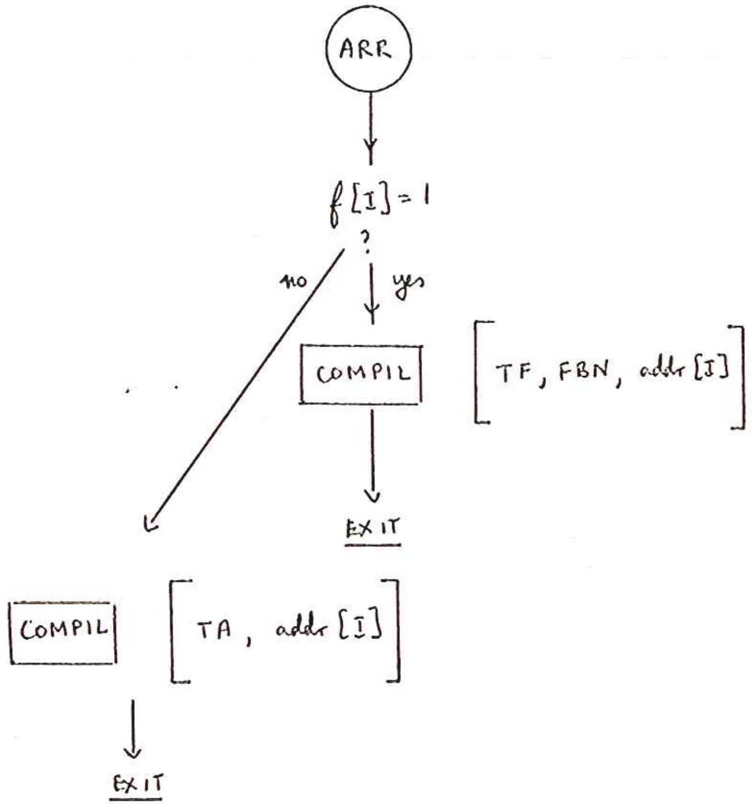
TAKID continued

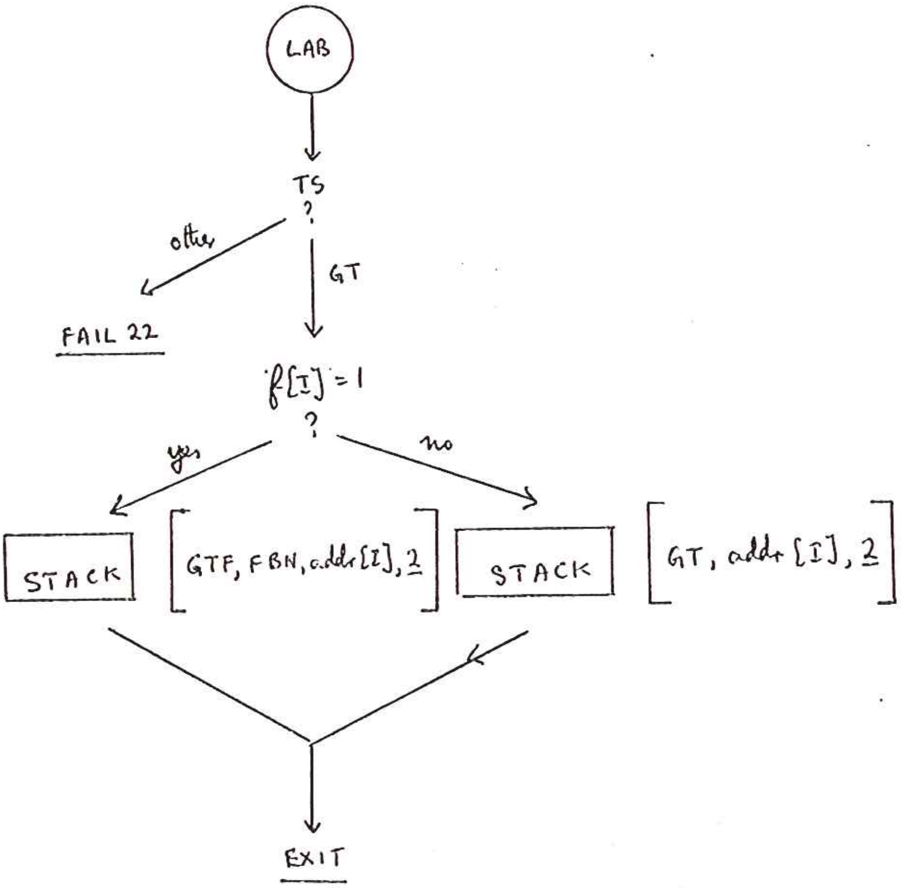
page 2 of 5



Fail if assignment to function name outside procedure body

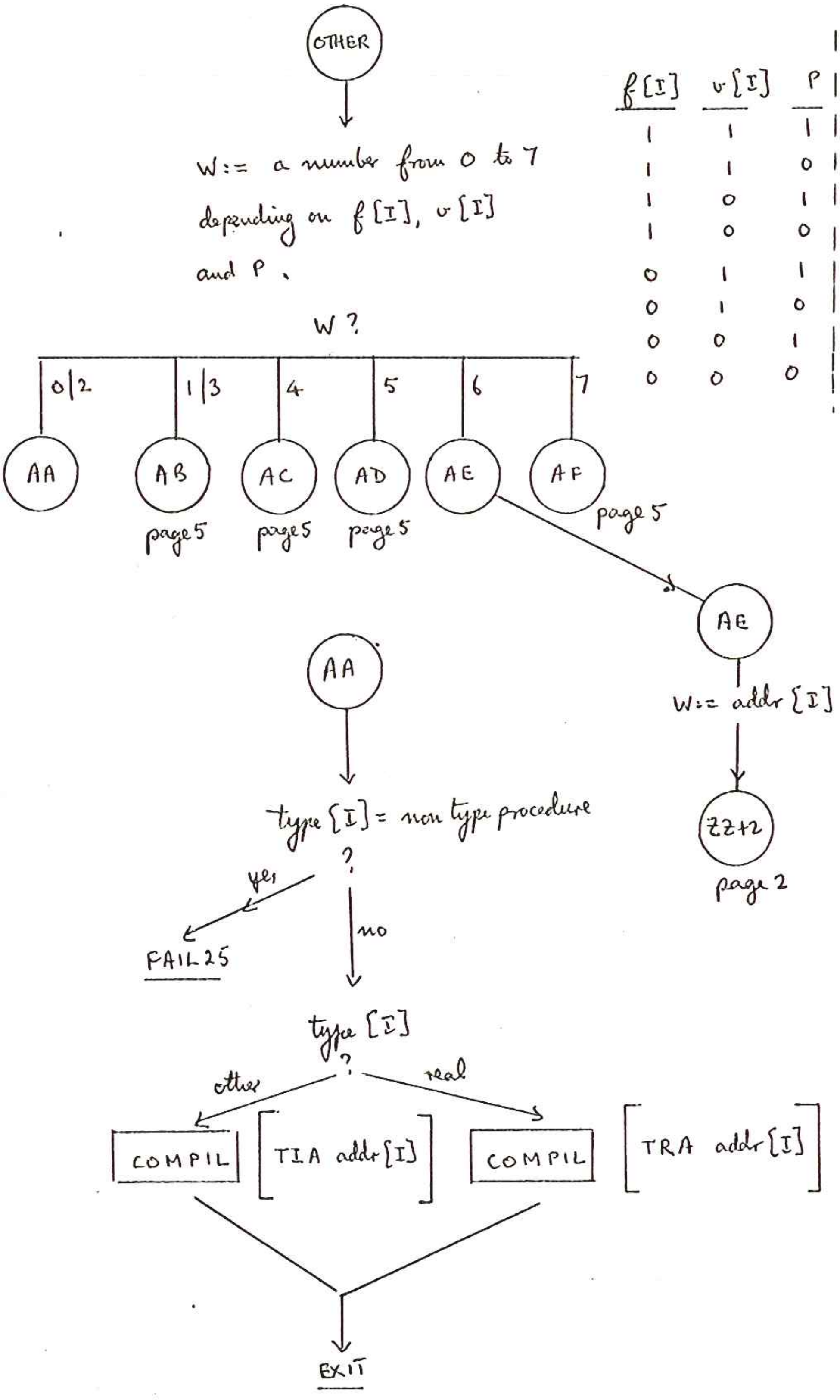
TAKID continued





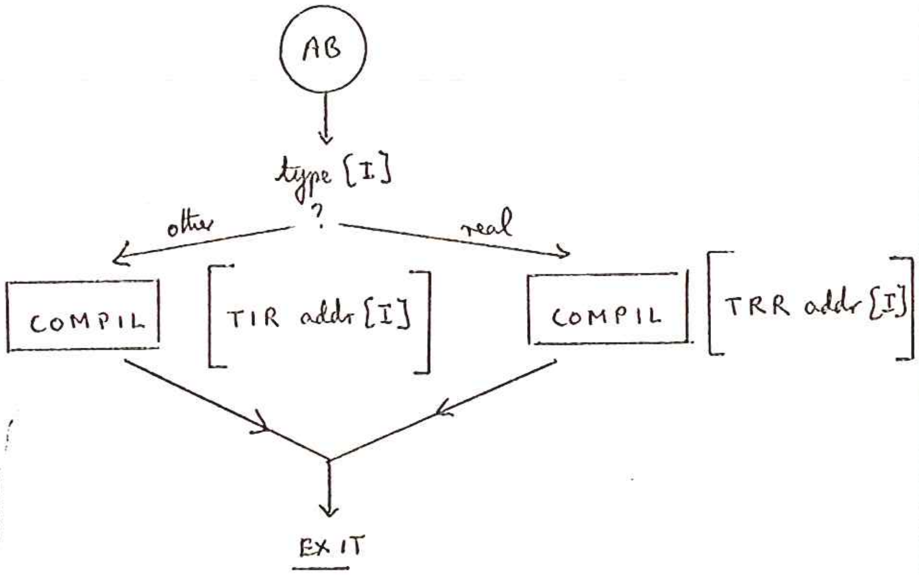
TAKID continued

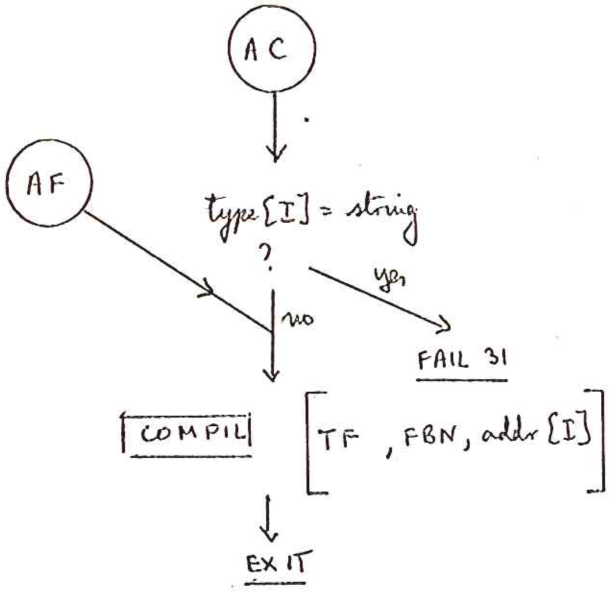
page 4 of 5

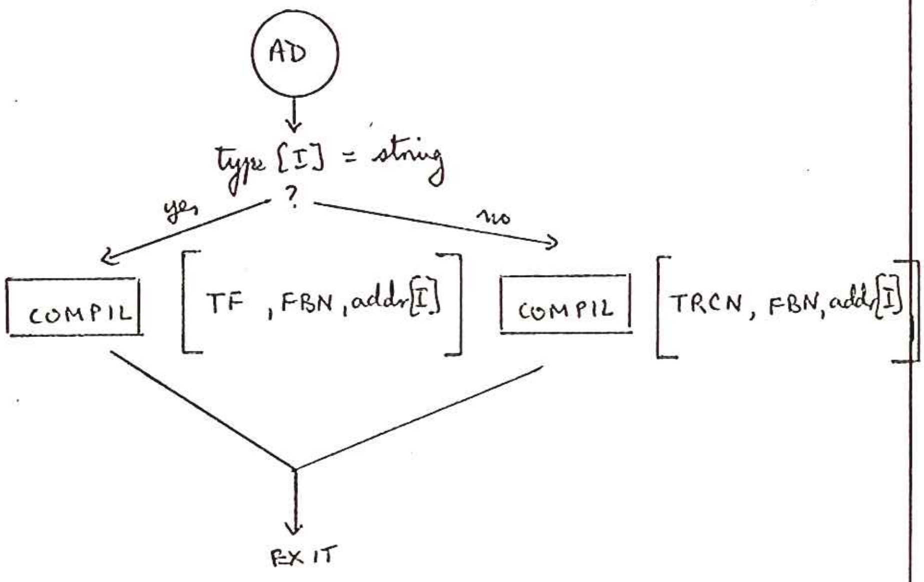


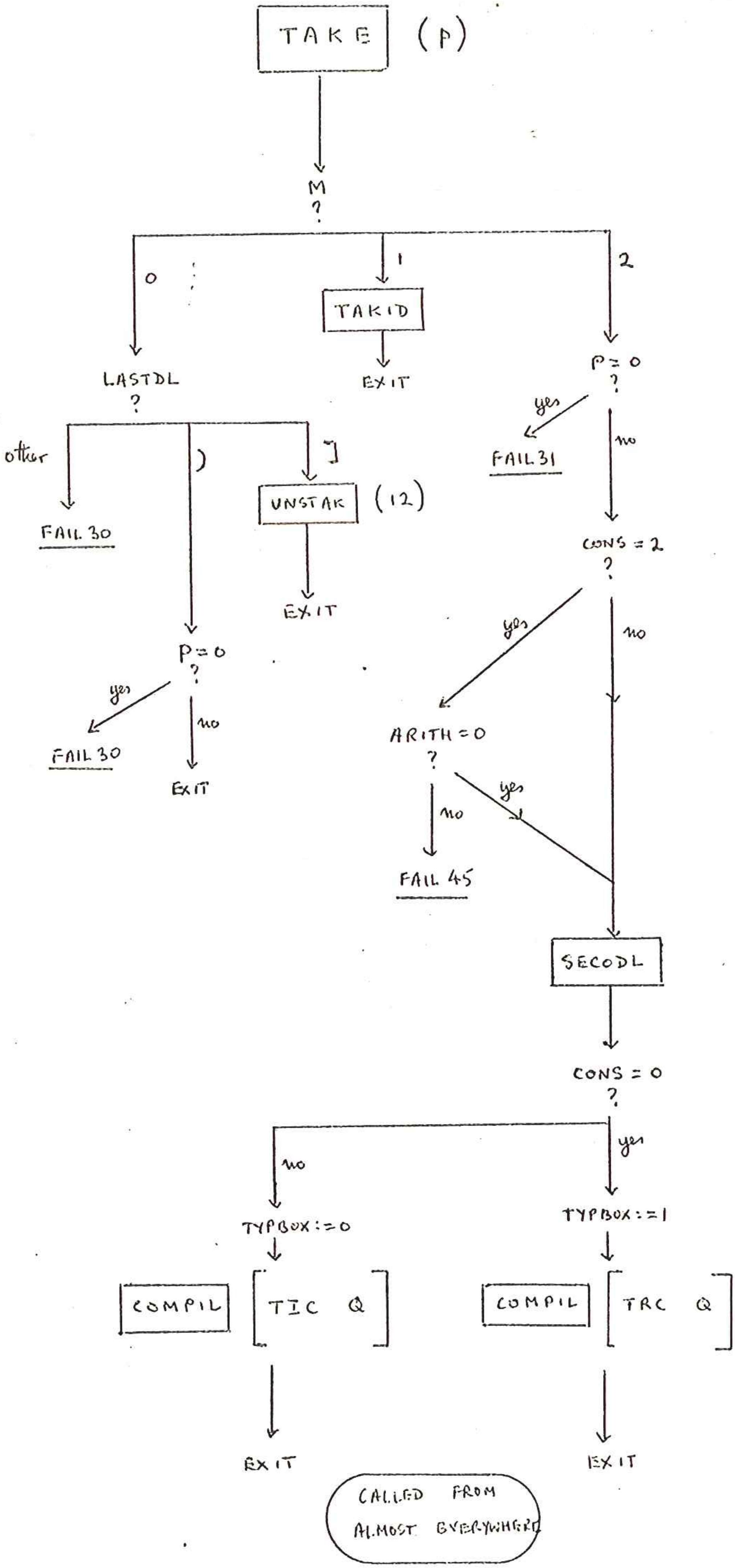
f[I]	v[I]	P	W
1	1	1	7
1	1	0	6
1	0	1	5
1	0	0	4
0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0

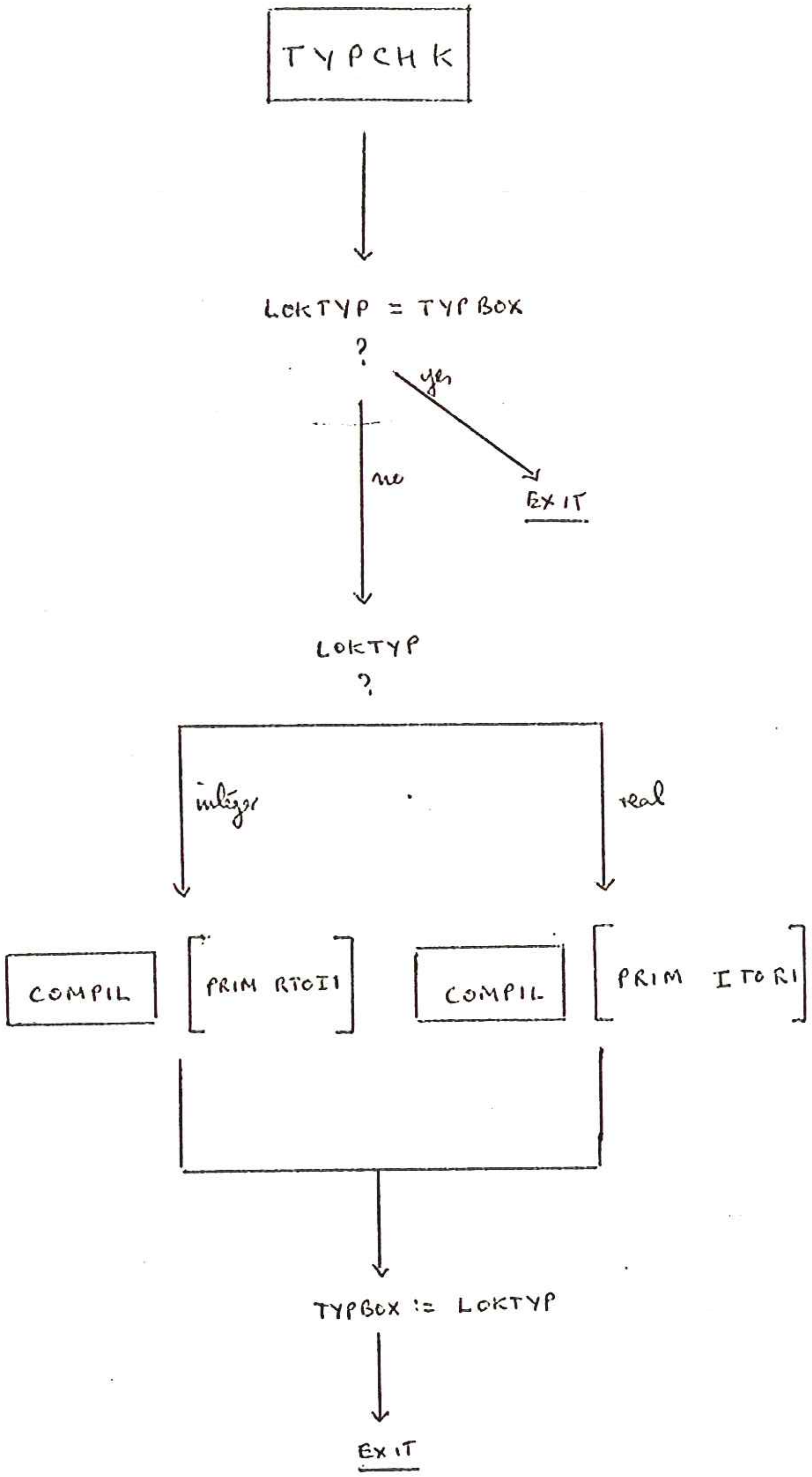
TAKID continued





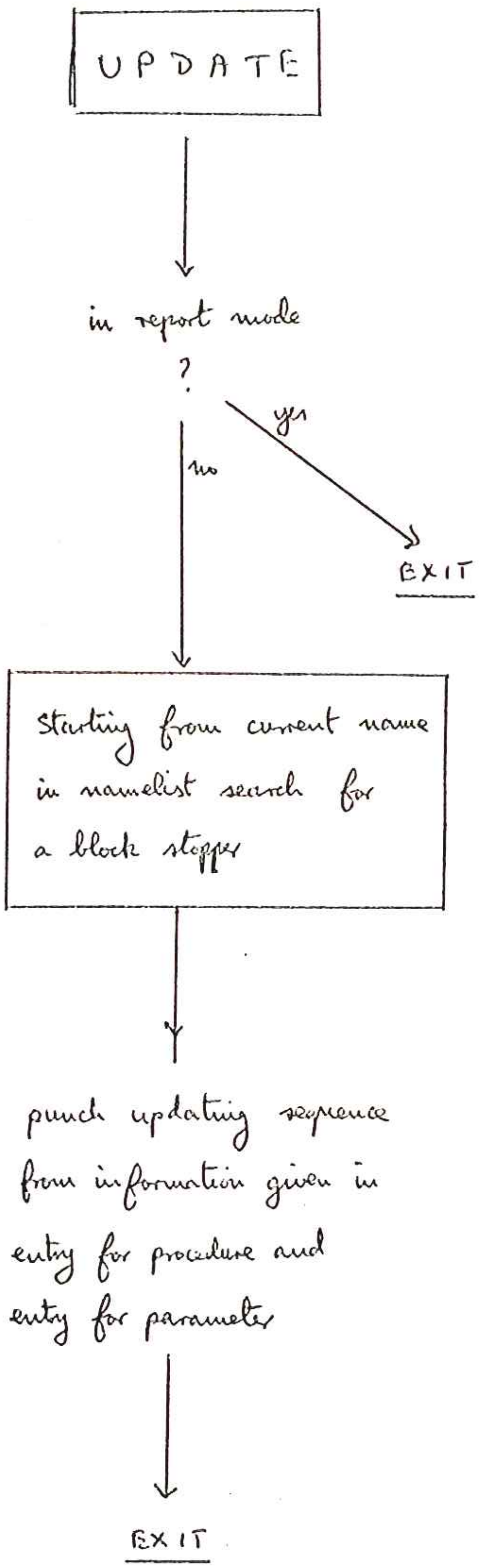



TAKE (P)
M ?
0
1
2
TAKID
EXIT
LASTDL ?
other
)
]
FAIL 30
UNSTAK (12)
EXIT
P=0 ?
yes
no
FAIL 30
EXIT
P = 0 ?
yes
FAIL 31
no
CONS = 2 ?
yes
no
ARITH = 0 ?
no
yes
FAIL 45
SECODL
CONS = 0 ?
no
yes
TYPBOX:=0
TYPBOX:=1
COMPIL
TIC Q
COMPIL
TRC Q
EXIT
EXIT
CALLED FROM ALMOST EVERYWHERE



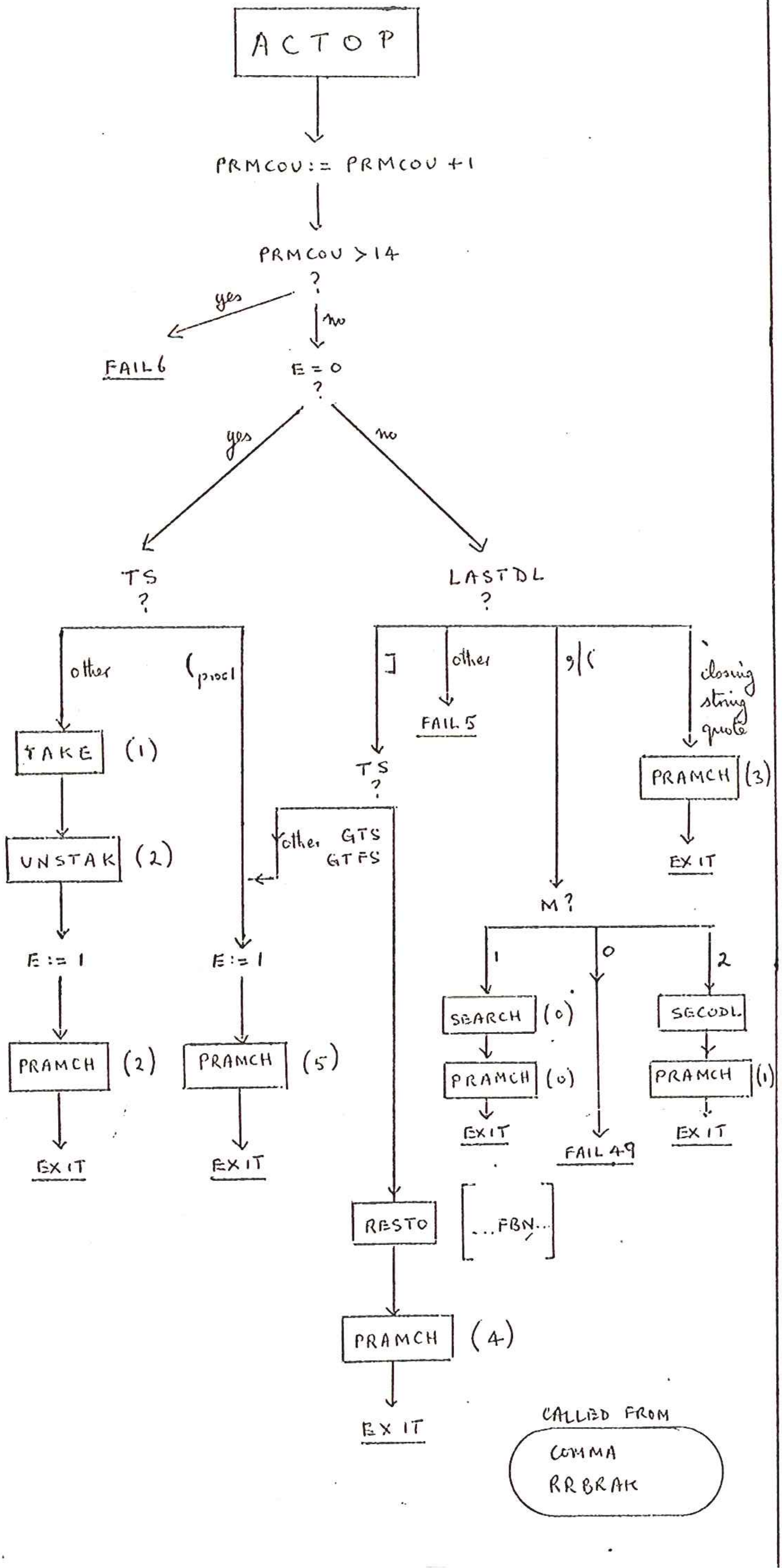
CALLED FROM

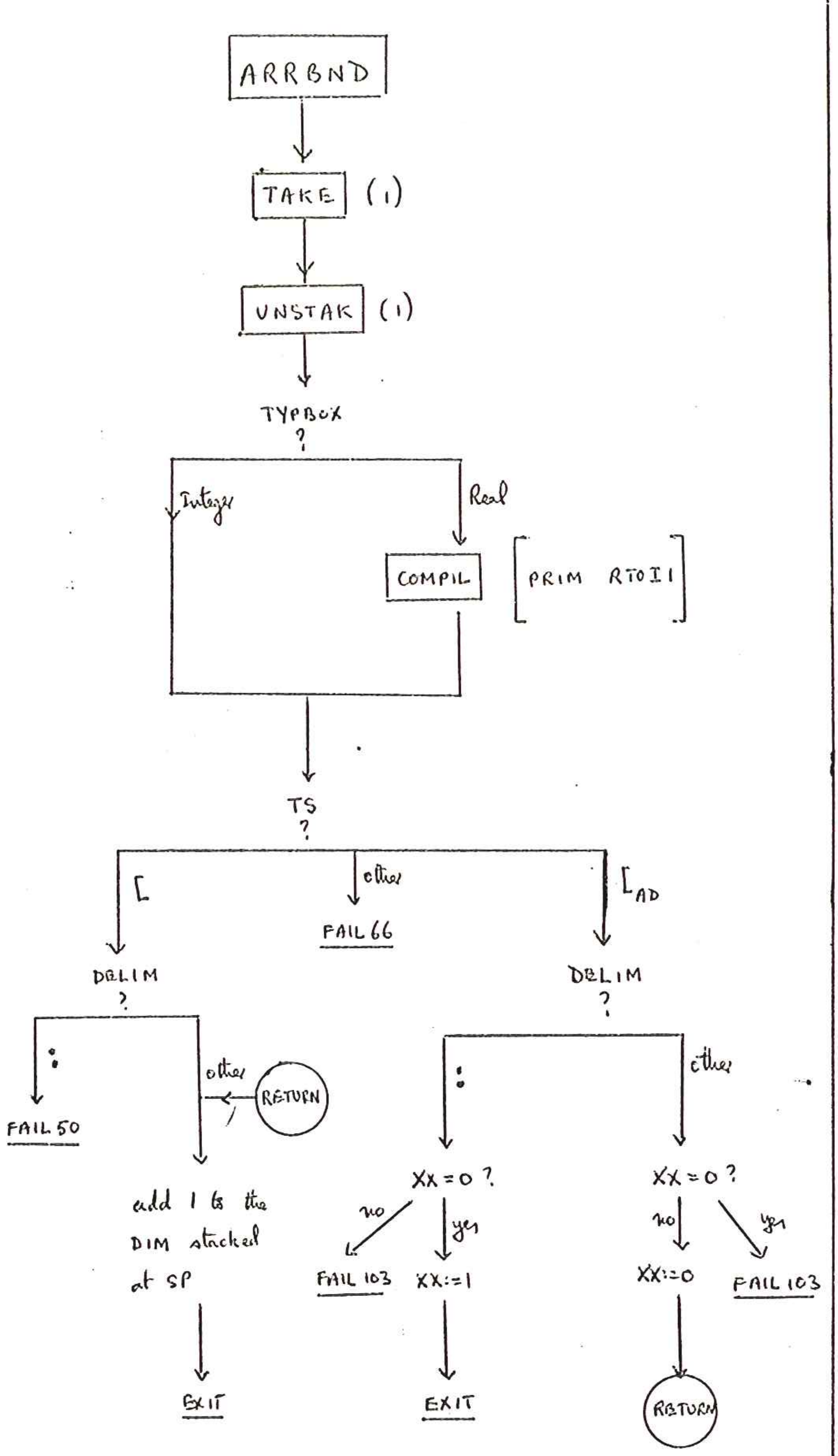
STEP FORCOM


UPDATE
in report mode
?
yes
no
EXIT
Starting from current name in namelist search for a block stopper
punch updating sequence from information given in entry for procedure and entry for parameter
EXIT

CALLED FROM

RSBRAK
RRBRAK
FDMCOM

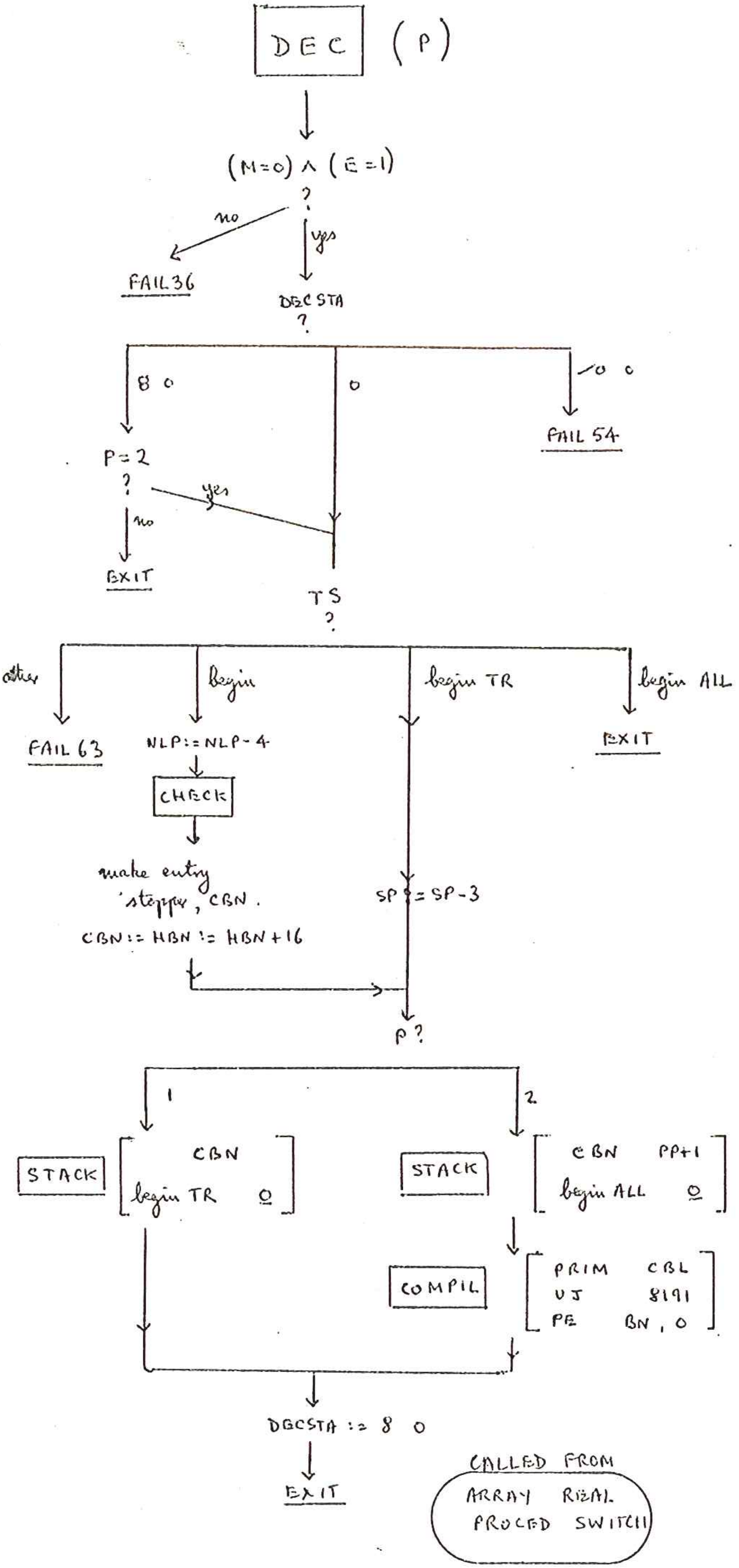

ACTOP
PRMCOU:= PRMCOU +1
PRMCOU >14
?
yes
no
FAIL6
E = 0
?
yes
no
TS
?
other
(proc
TAKE
(1)
UNSTAK
(2)
E:= 1
PRAMCH
(2)
EXIT
E:= 1
PRAMCH
(5)
EXIT
LASTDL
?
]
other
FAIL 5
;|(
closing string quote
PRAMCH
(3)
EXIT
TS
?
other
GTS
GTFS
RESTO
...FBN...
PRAMCH
(4)
EXIT
M ?
1
0
2
SEARCH
(0)
PRAMCH
(0)
EXIT
FAIL 49
SECODL
PRAMCH
(1)
EXIT
CALLED FROM
COMMA
RRBRAK



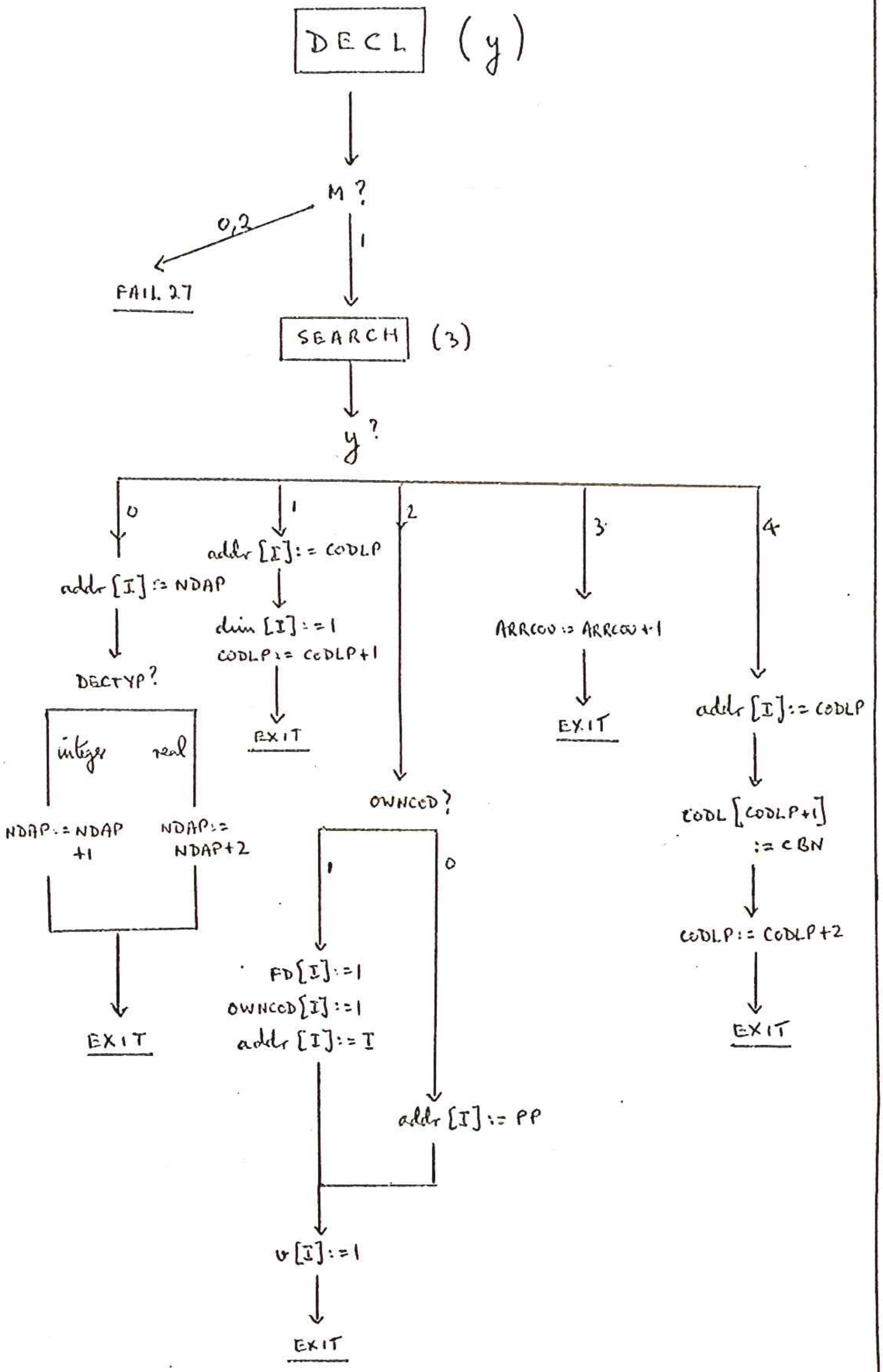
XX is the bit stored in the top of the stack

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

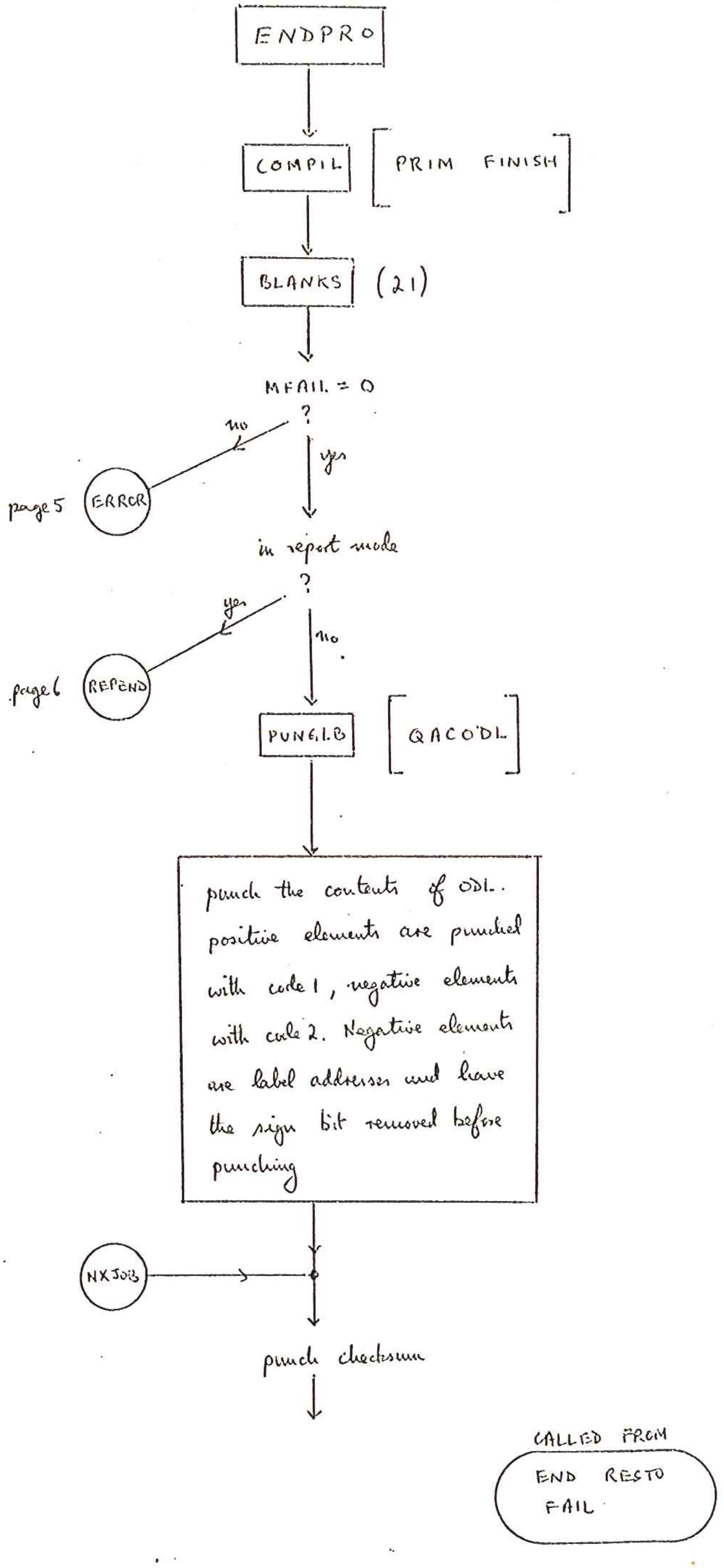
FD set to avoid fail in FCLAPS

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

CALLED FROM

REAL PROCED
SWITCH SEMICO
LSBRAK COMMA FAIL

page 1 of 6

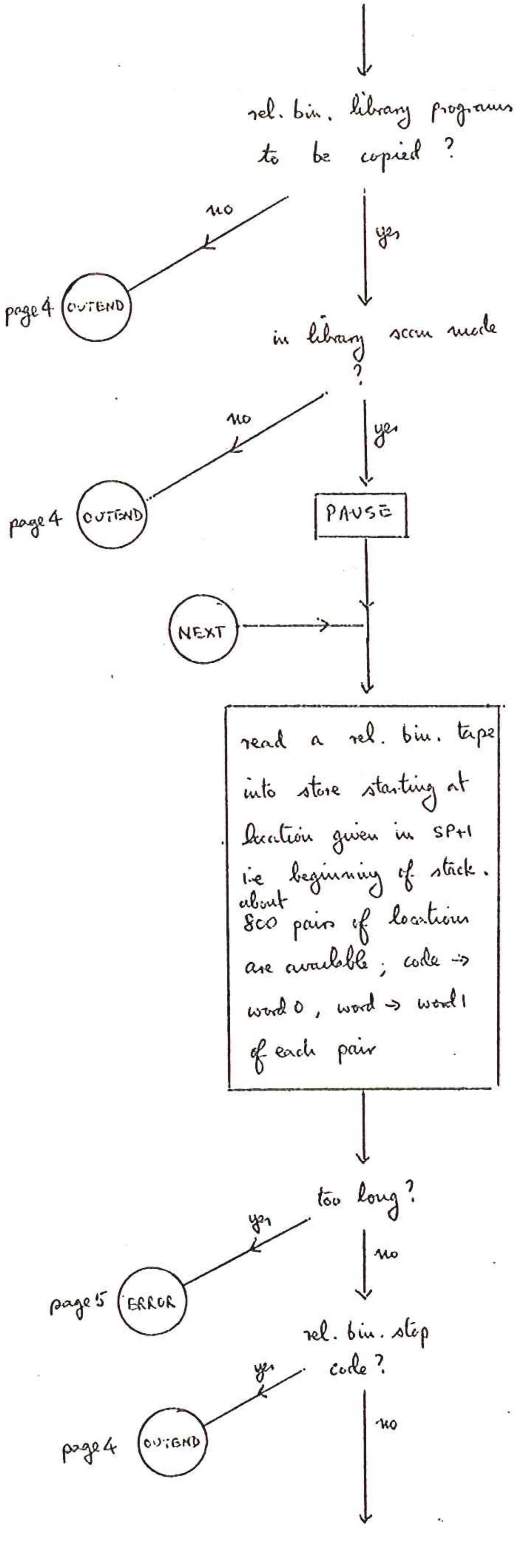


punch the contents of ODL. positive elements are punched with code 1, negative elements with code 2. Negative elements are label addresses and have the sign bit removed before punching

in report mode if OPTION 2 bit present

punch global L H. label

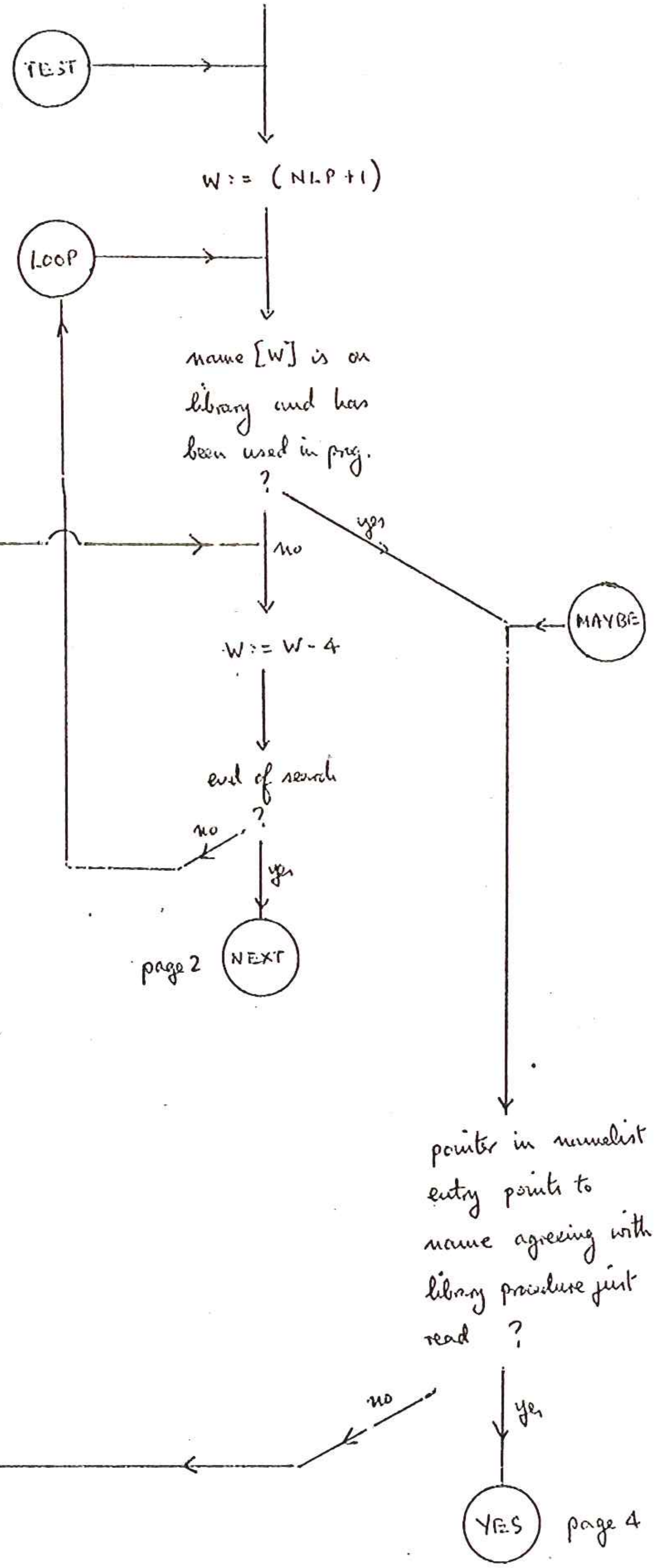
ENDPRO continued


rel. bin. library programs to be copied ?
no
yes
page 4 OUTEND
in library scan mode ?
no
yes
page 4 OUTEND
PAUSE
NEXT
read a rel. bin. tape into store starting at location given in SP+1 ie beginning of stack. about 800 pairs of locations are available ; code → word 0 , word → word 1 of each pair
too long ?
yes
no
page 5 ERROR
rel. bin. stop code ?
yes
no
page 4 OUTEND

in library scan mode if OPTION +8 bit present

Restart at 9 by User

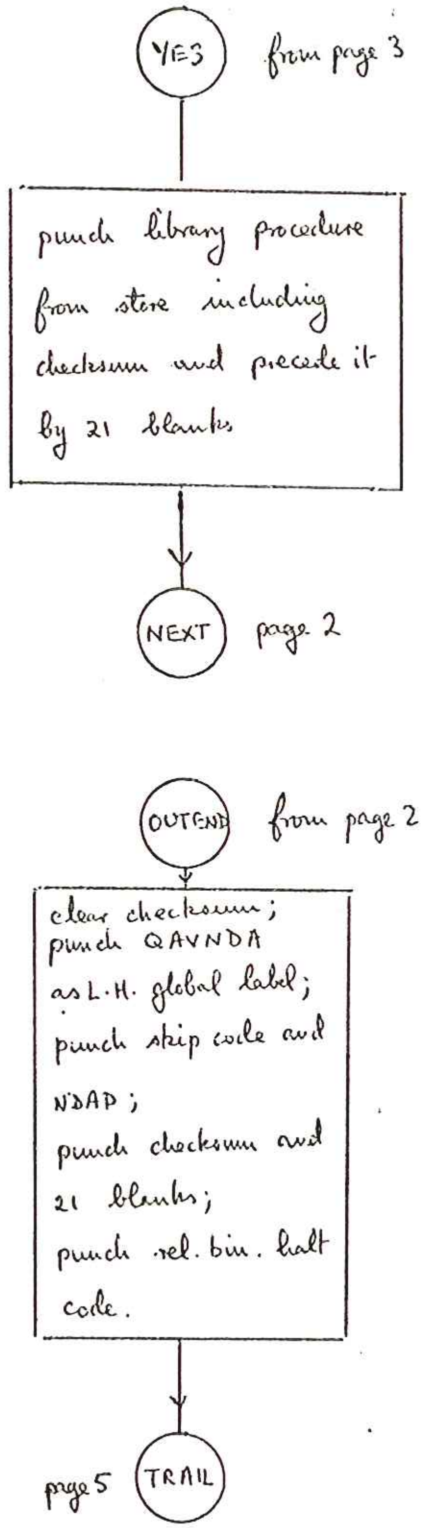
ENDPRO continued


TEST
W := (NLP+1)
LOOP
name [W] is on library and has been used in prog. ?
yes
no
MAYBE
W := W - 4
end of search ?
no
yes
page 2
NEXT
pointer in namelist entry points to name agreeing with library procedure just read ?
no
yes
YES
page 4

(NLP+1) points to top of namelist

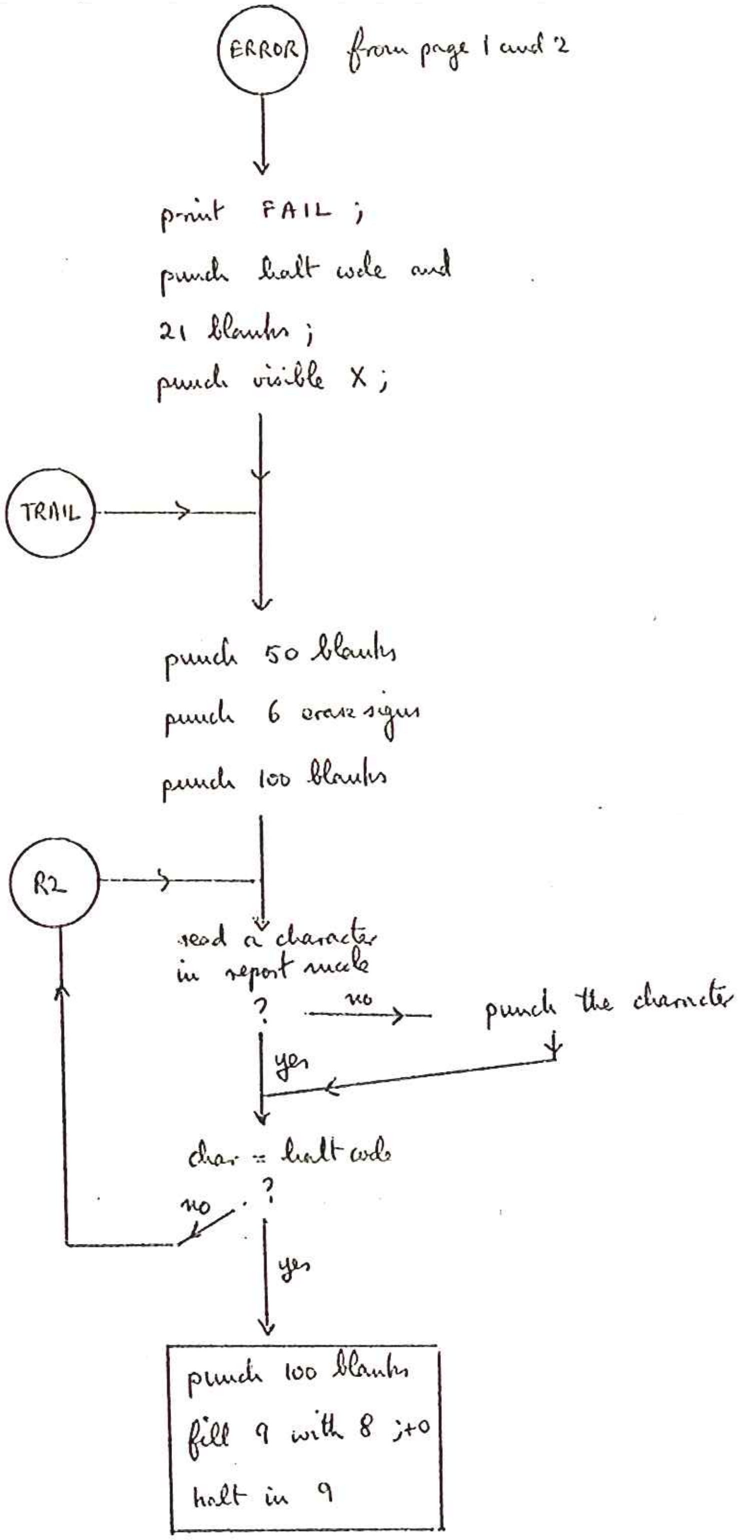
search ends at point including names declared in Users outermost block

ENDPRO continued


YES
from page 3
punch library procedure from store including checksum and precede it by 21 blanks
NEXT
page 2
OUTEND
from page 2
clear checksum;
punch QAVNDA as L.H. global label;
punch skip code and NDAP;
punch checksum and 21 blanks;
punch rel. bin. halt code.
page 5
TRAIL

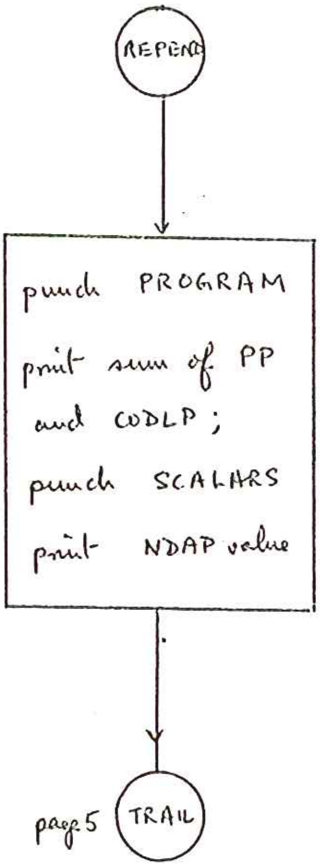
ENDPRO continued

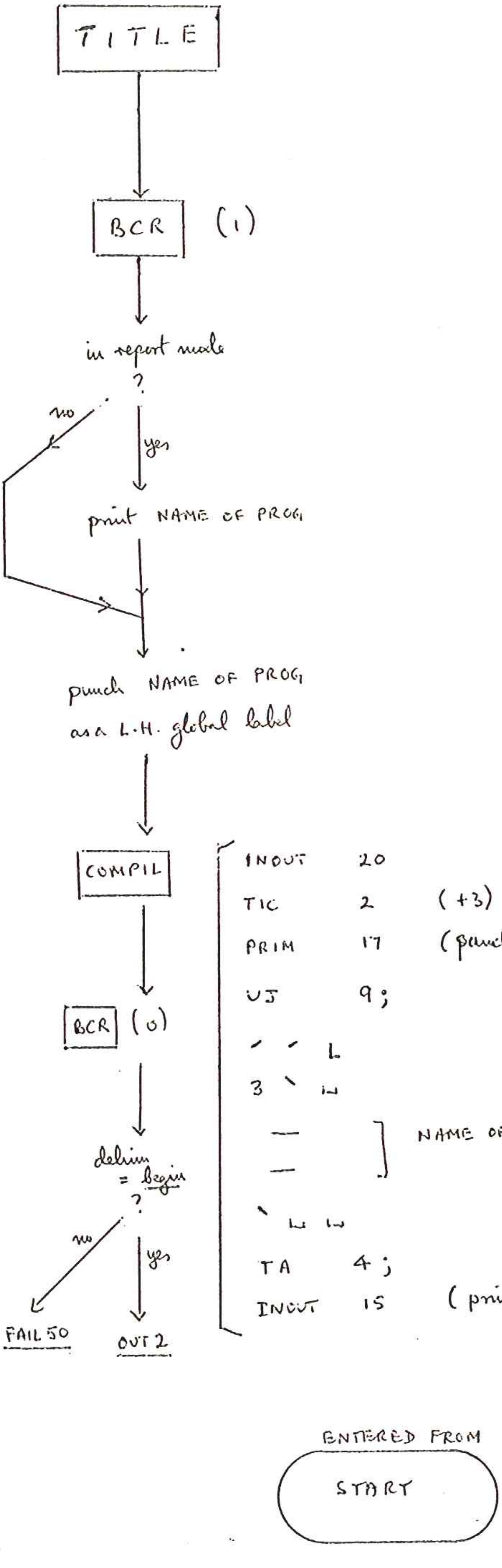
page 5 of 6



ENDPRO continued

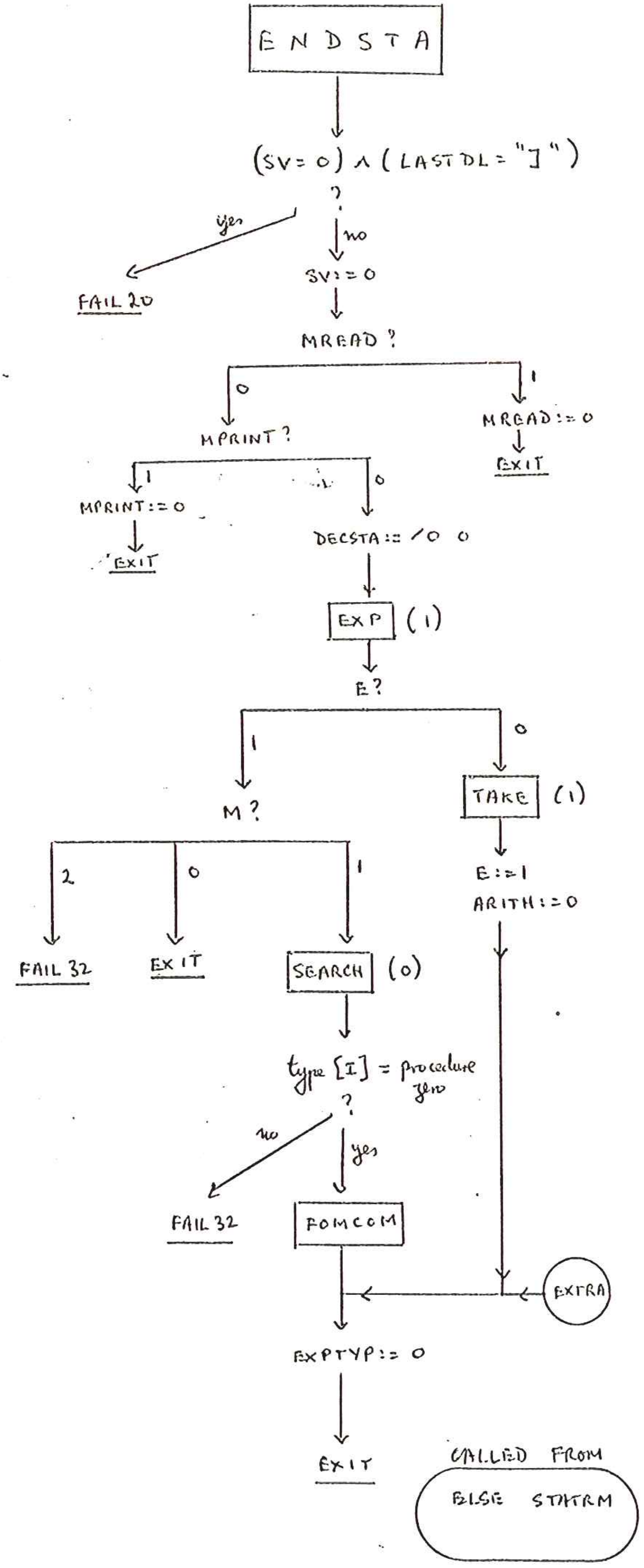
page 6 contlist

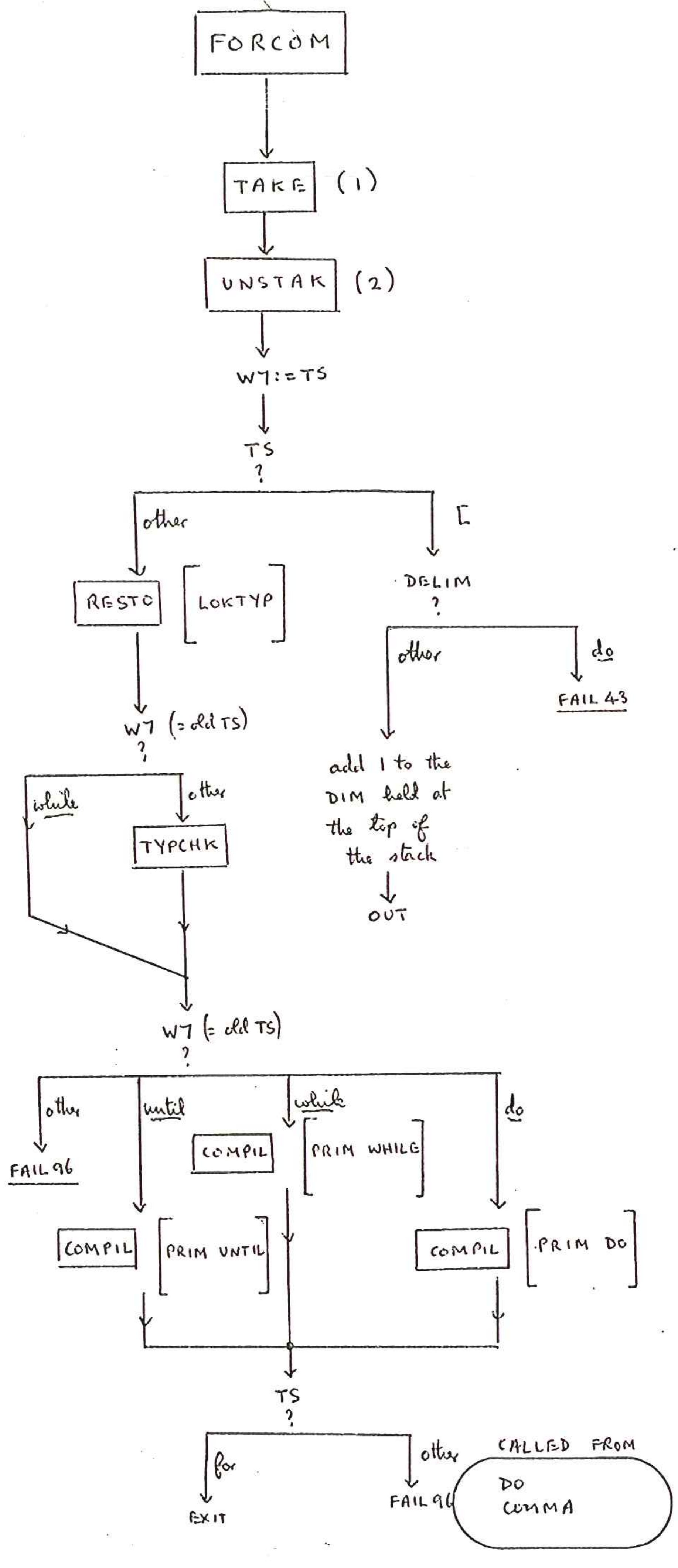




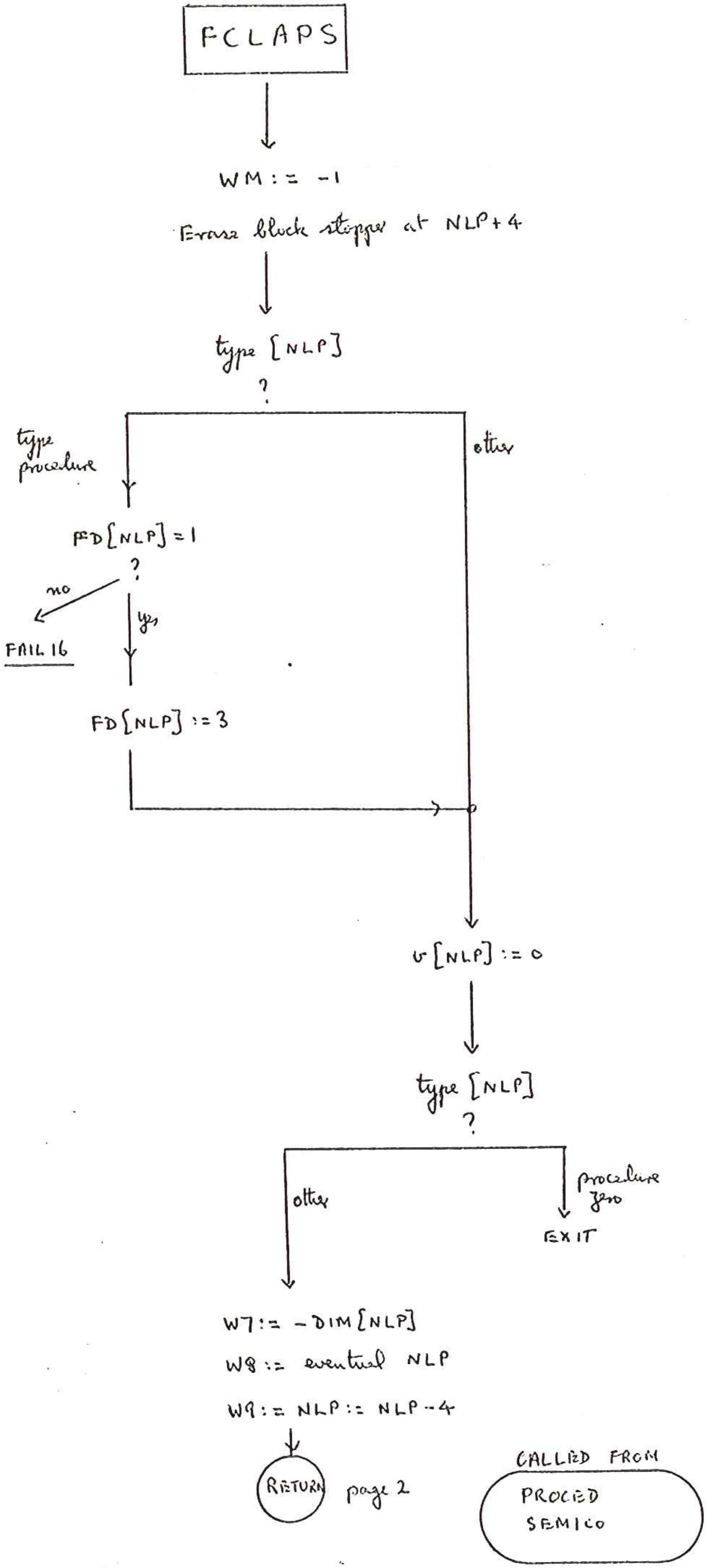
```
INOUT    20
TIC      2         (+3)
PRIM     17        (punch)
UJ       9;
´  ´  L
3  `  ␣
  —
            ]   NAME OF PROG
  —
`  ␣  ␣
TA       4;
INOUT    15        (print string)
```

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


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


FORCOM
TAKE (1)
UNSTAK (2)
W7:=TS
TS ?
other
[
RESTC
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

page 1 of 2

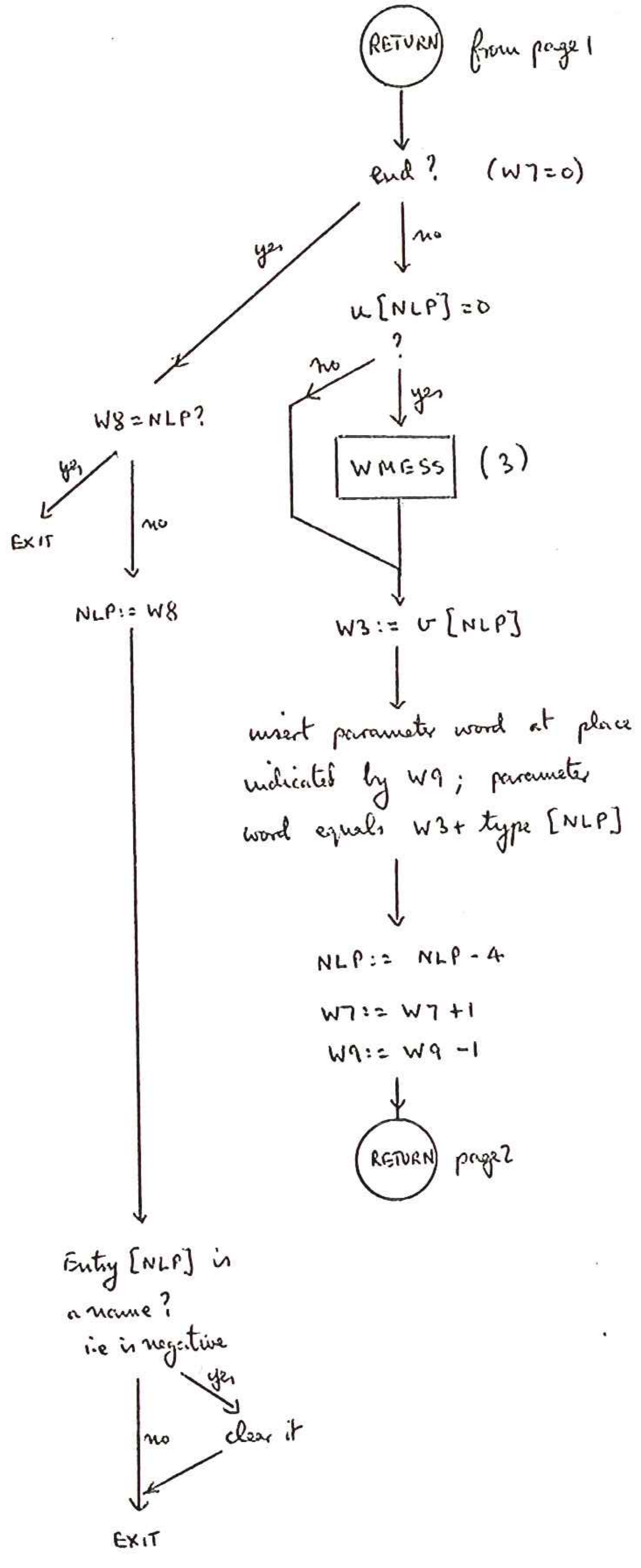


v bit = 1 during proc. body - used to detect recursion

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

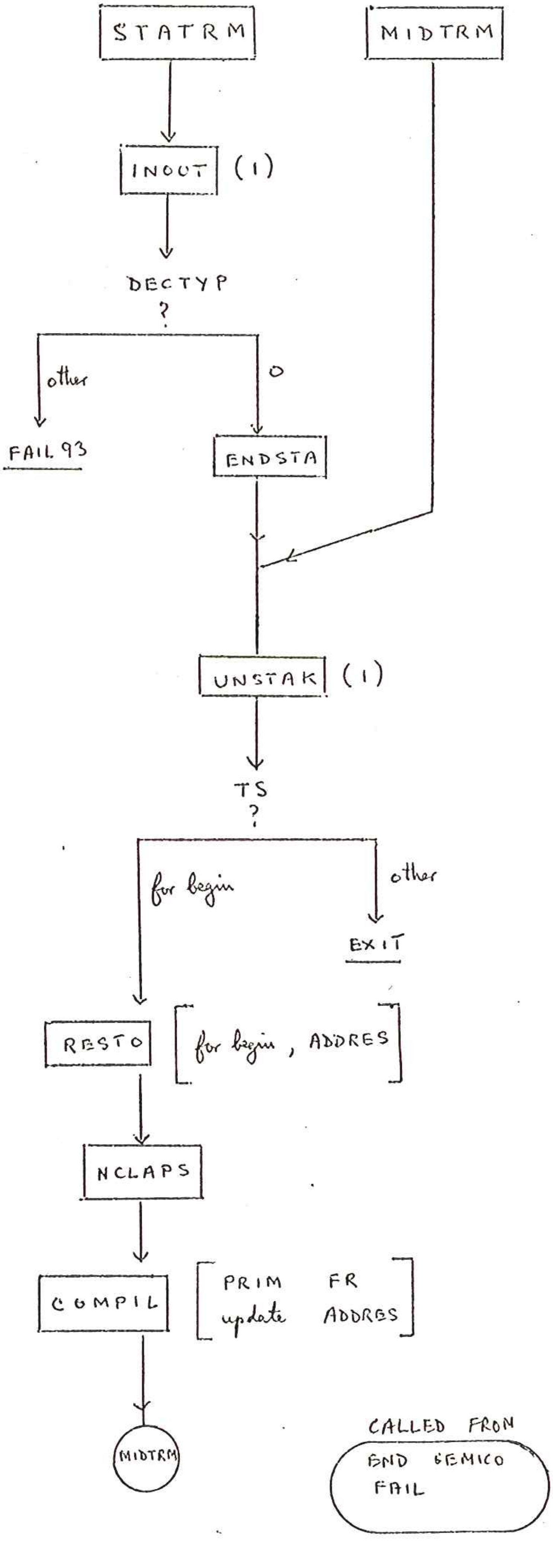
FCLAPS continued

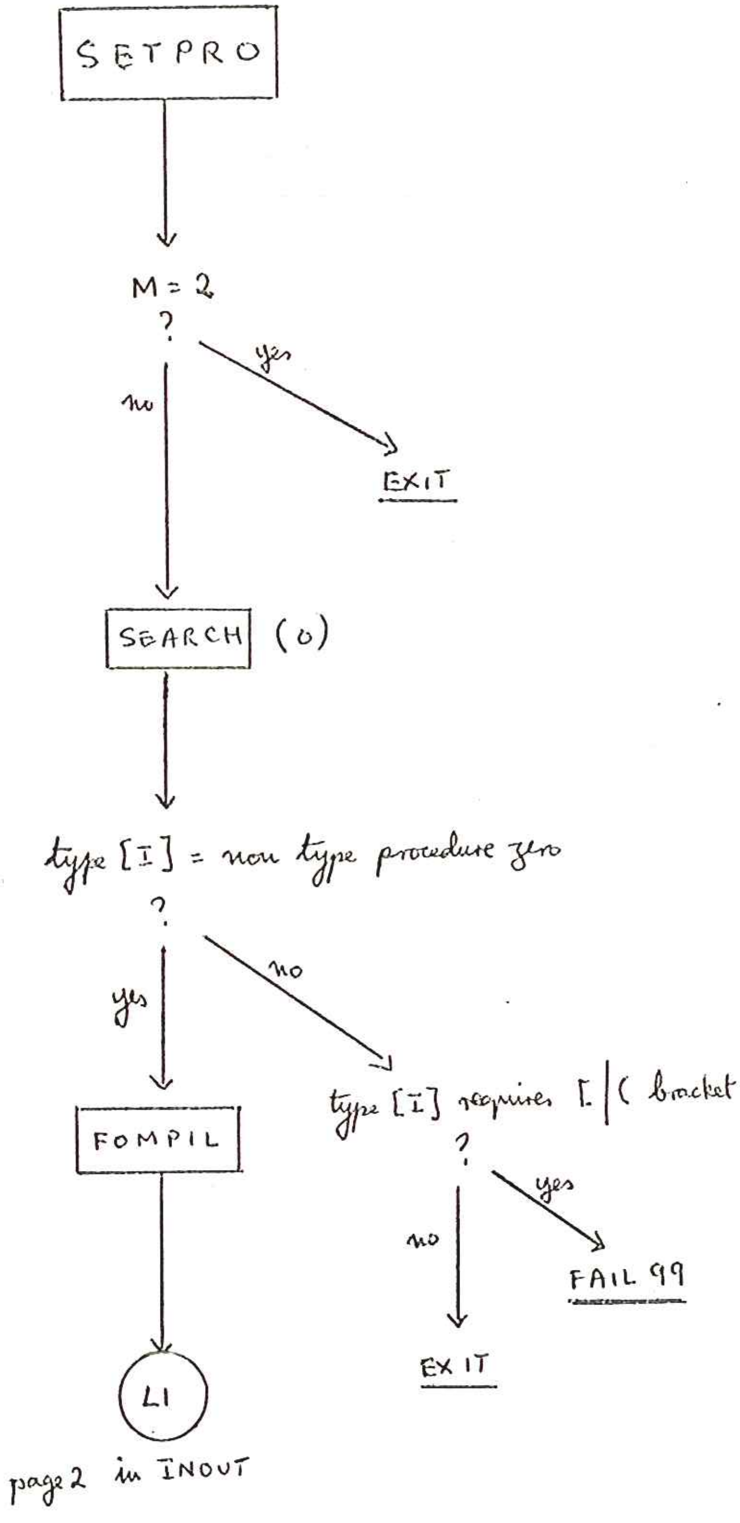
page 2 of 2



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.

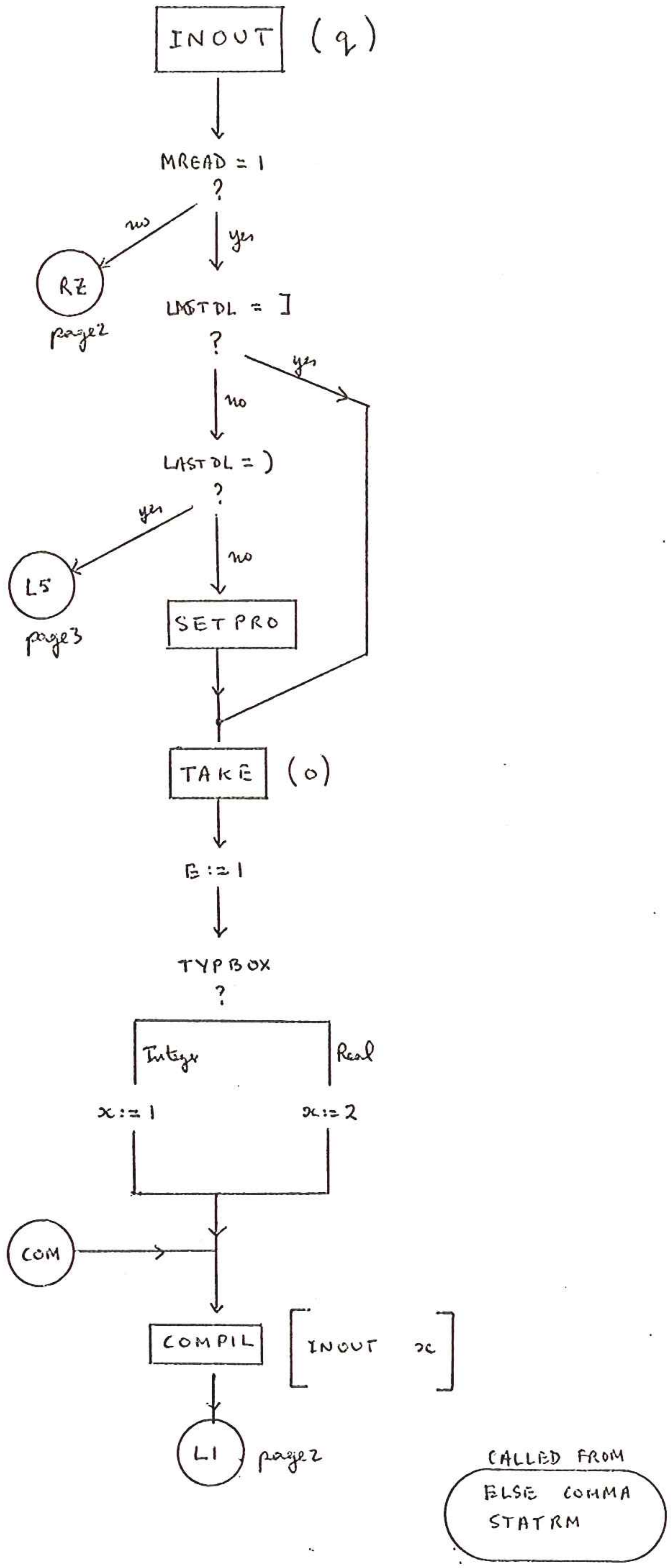

STATRM
MIDTRM
INOUT (1)
DECTYP ?
other
0
FAIL 93
ENDSTA
UNSTAK (1)
TS ?
for begin
other
EXIT
RESTO
[for begin , ADDRES]
NCLAPS
COMPIL
[PRIM FR update ADDRES]
MIDTRM
CALLED FROM
END SEMICO
FAIL



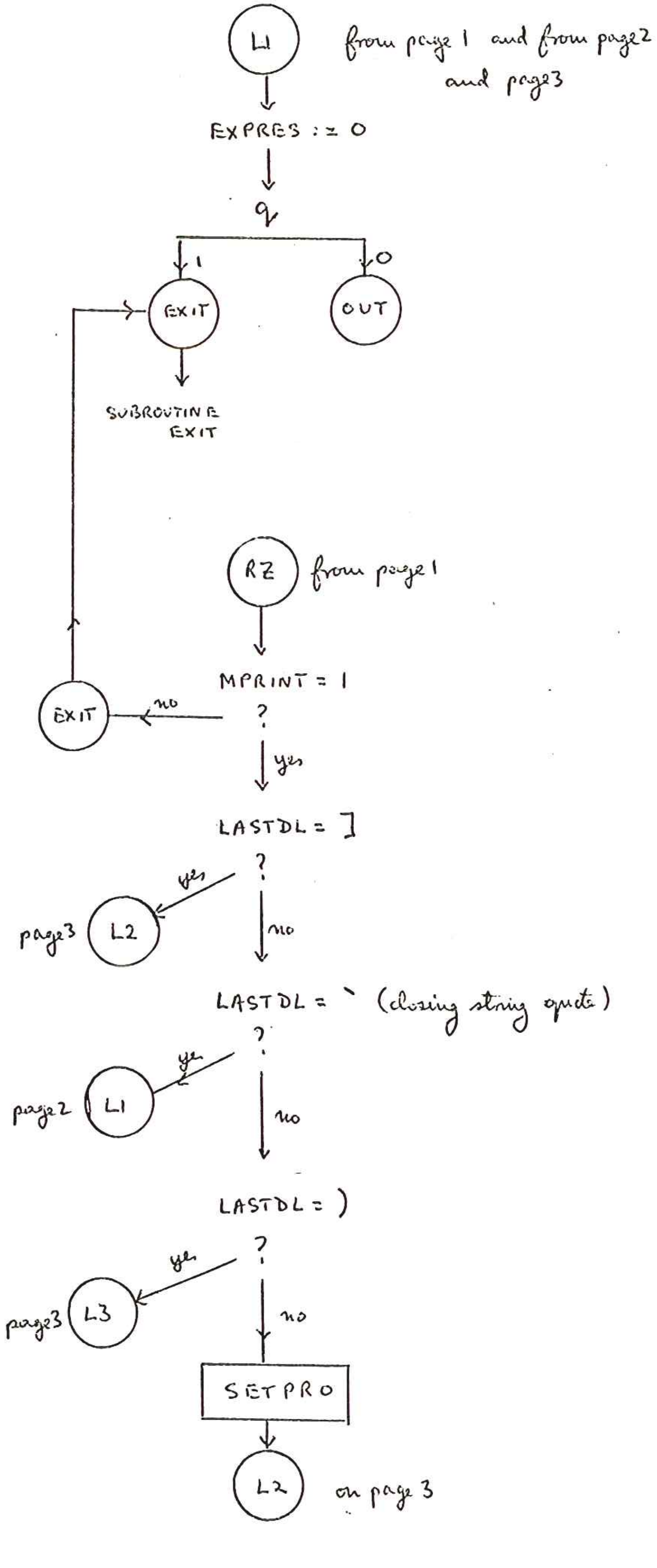
CALLED FROM INOUT

page 1 of 3

q is stored in W+11

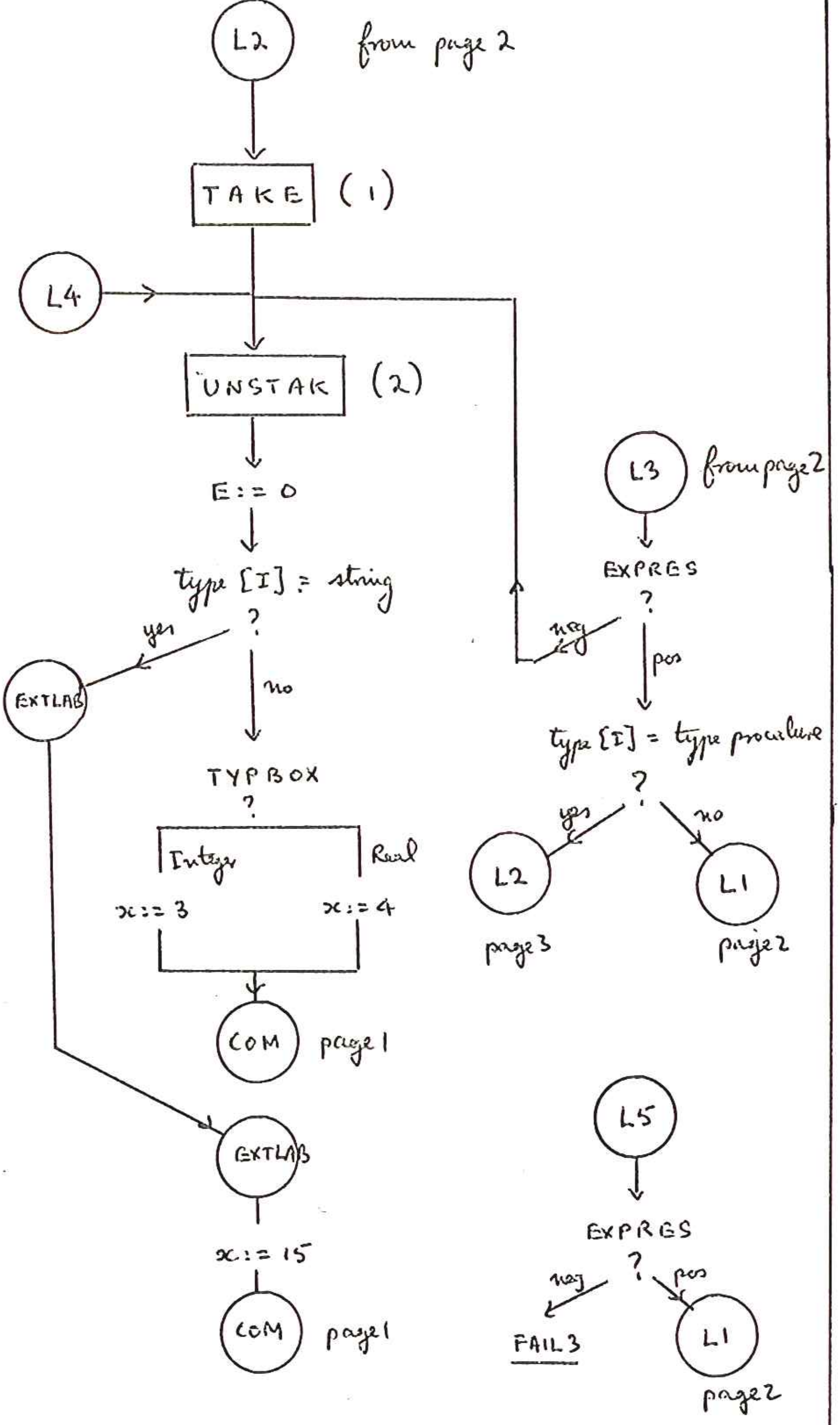


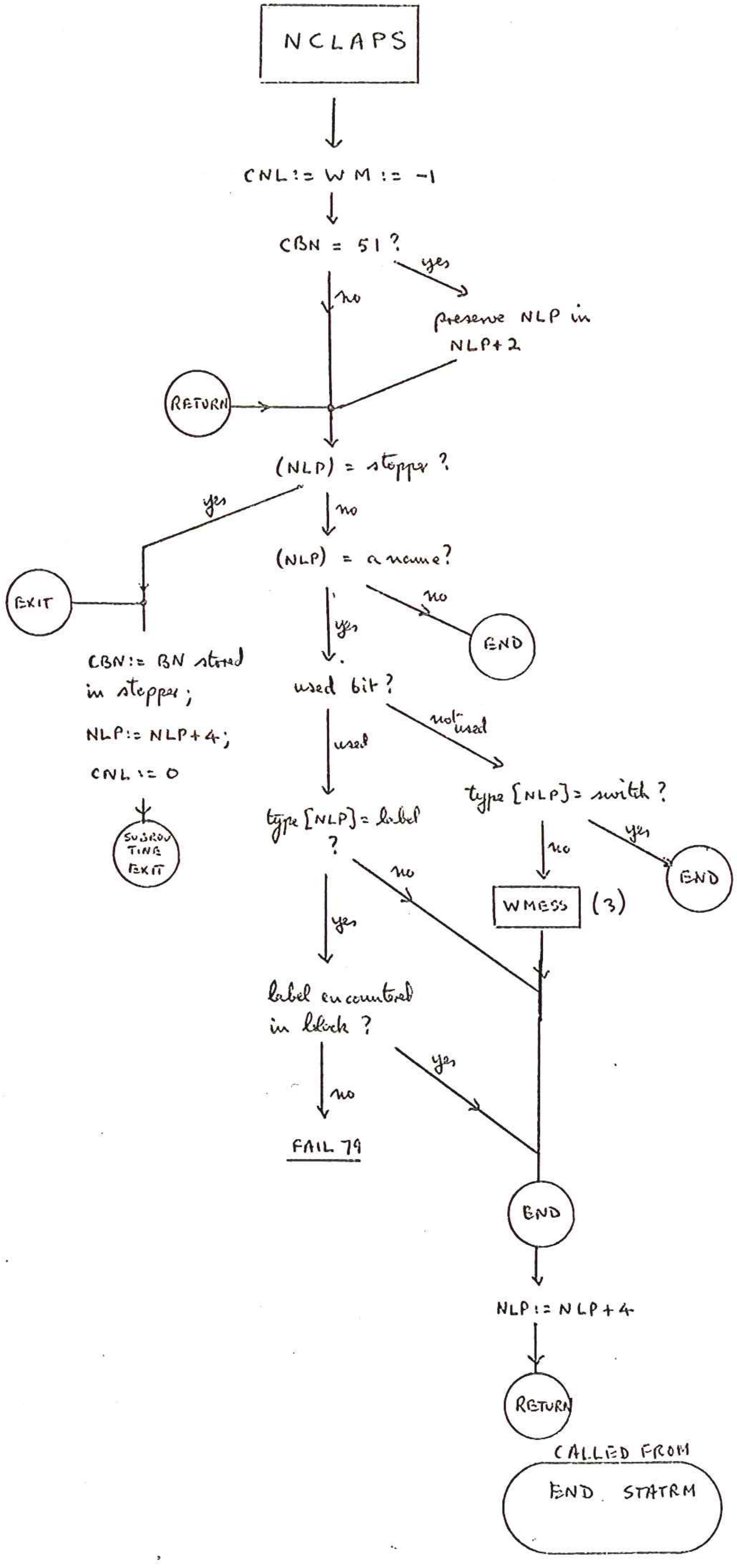
INOUT (q) continued.



INOUT (9) continued

page 3 of 3





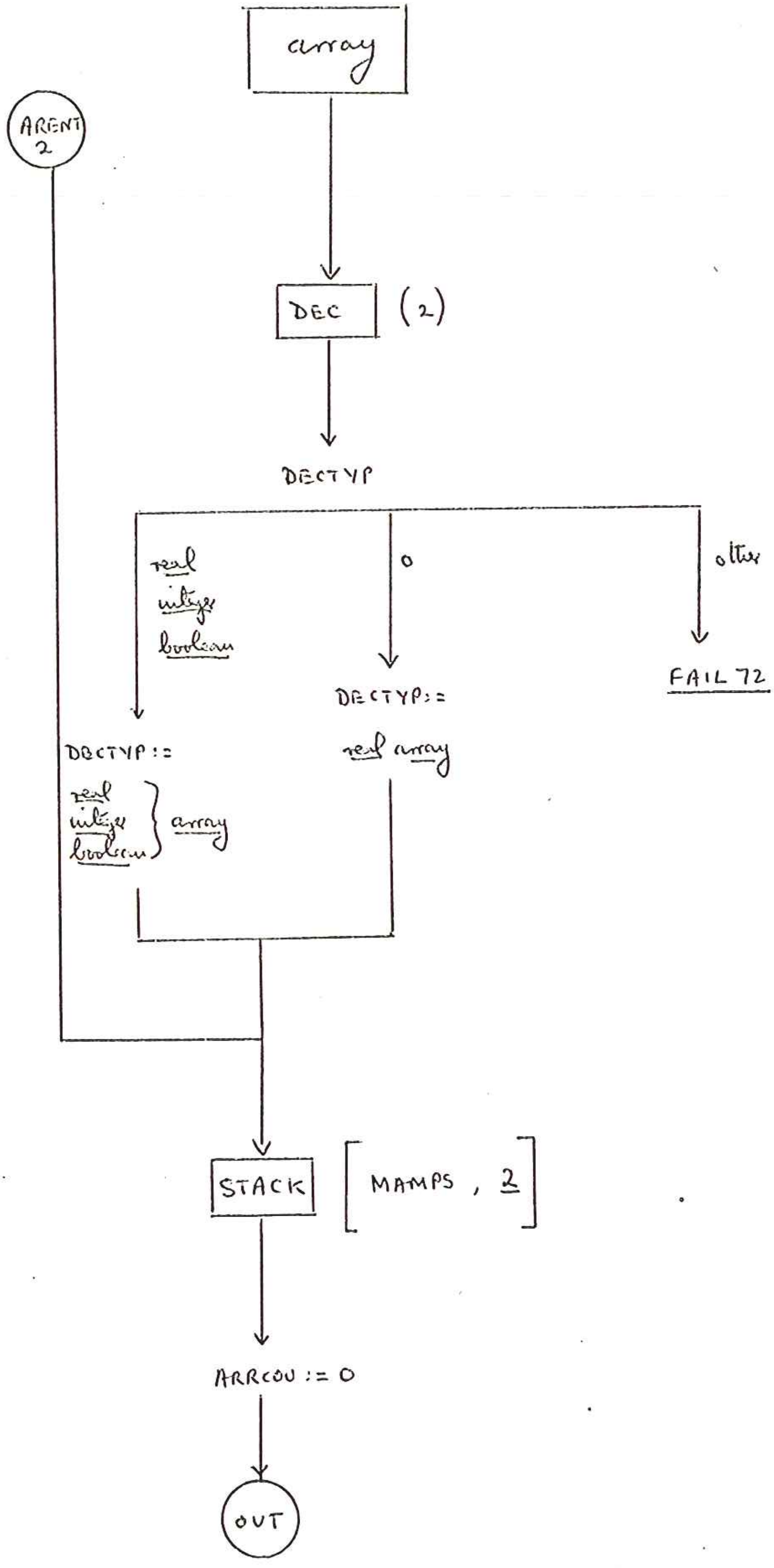
preservation purpose is so that library scan can include

calc
algol } procedures

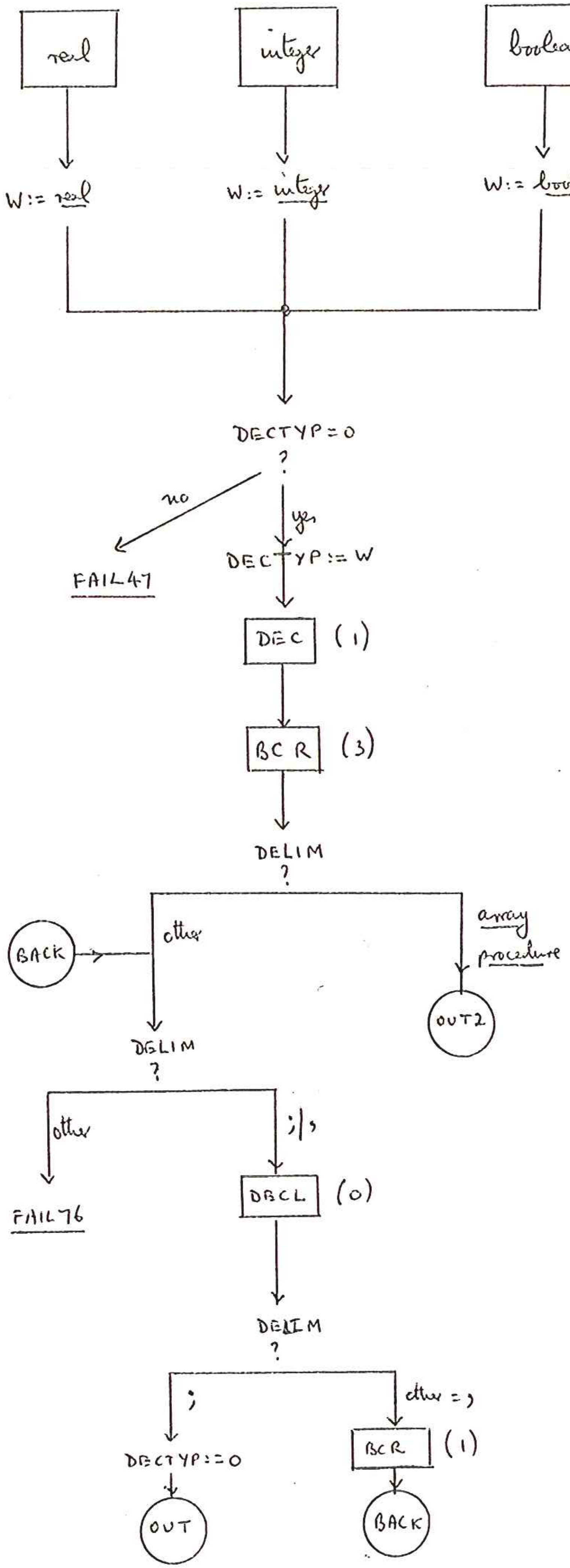
stopper is -1

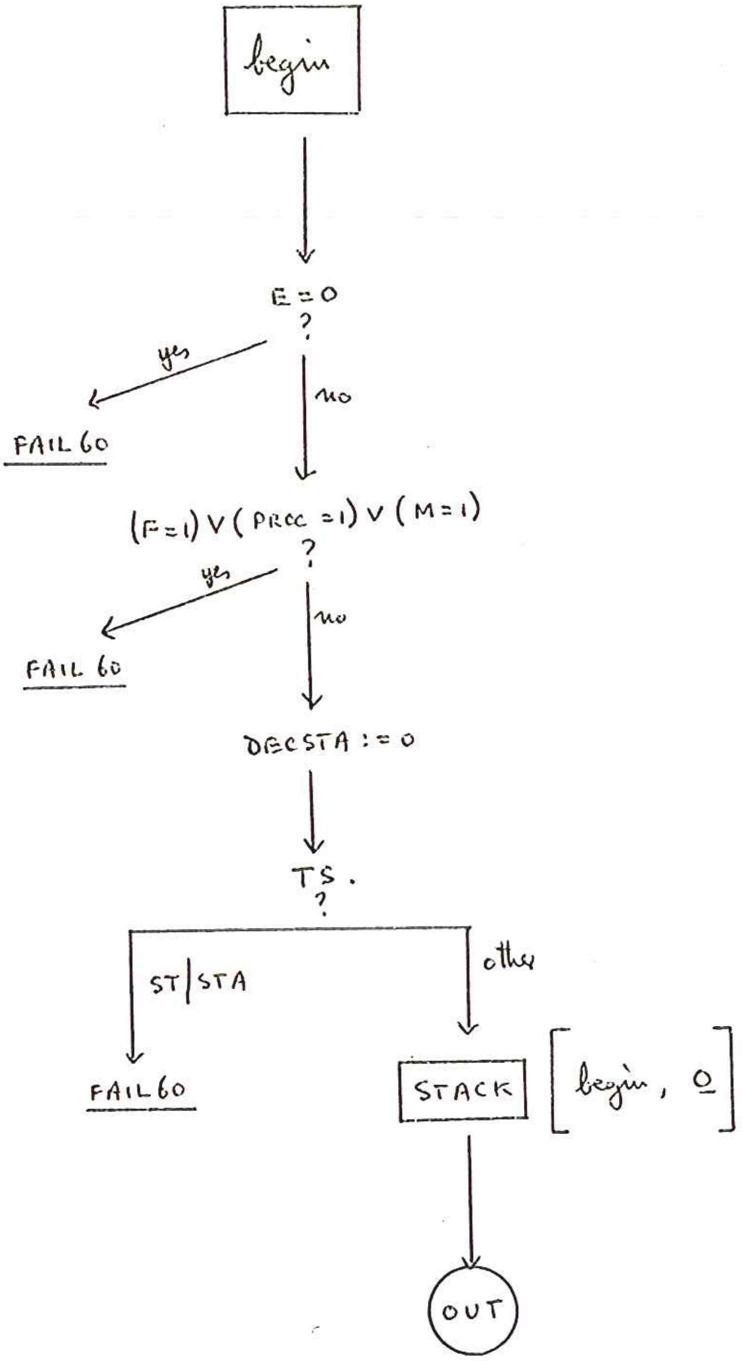
(NLP) is negative for a name

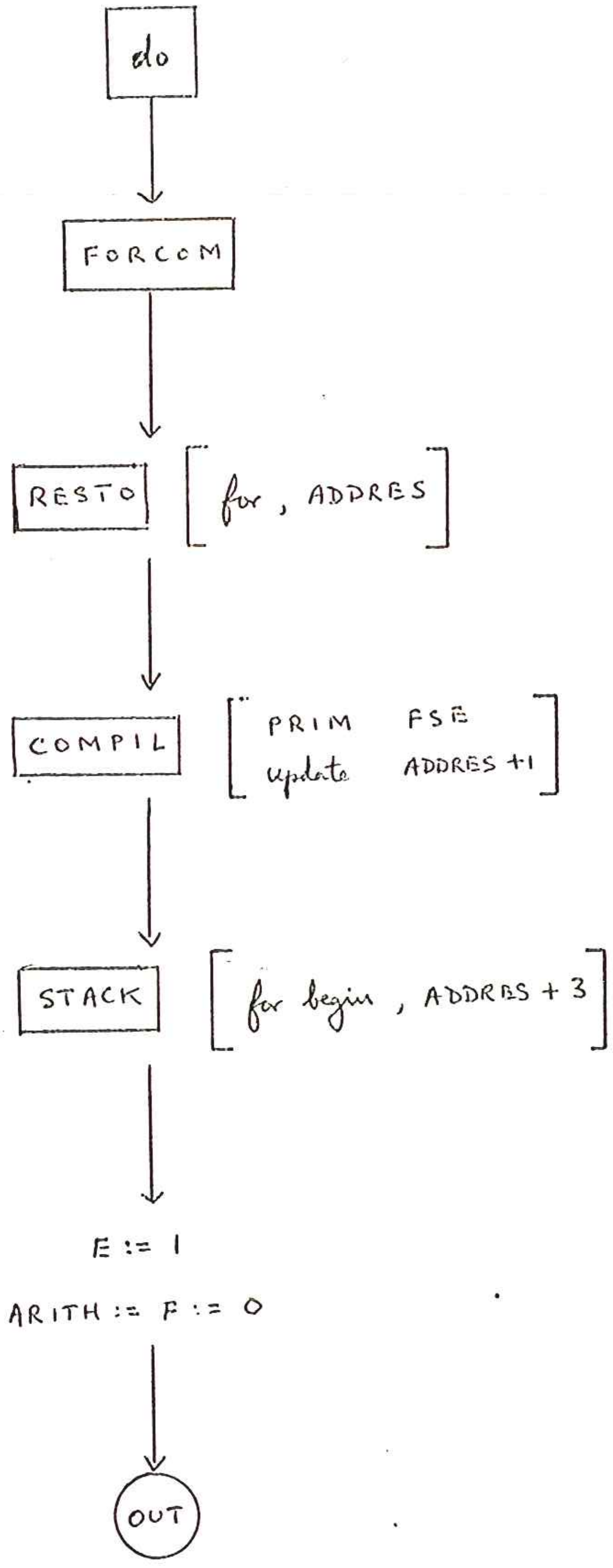
examine CODL entry: non zero if encountered.

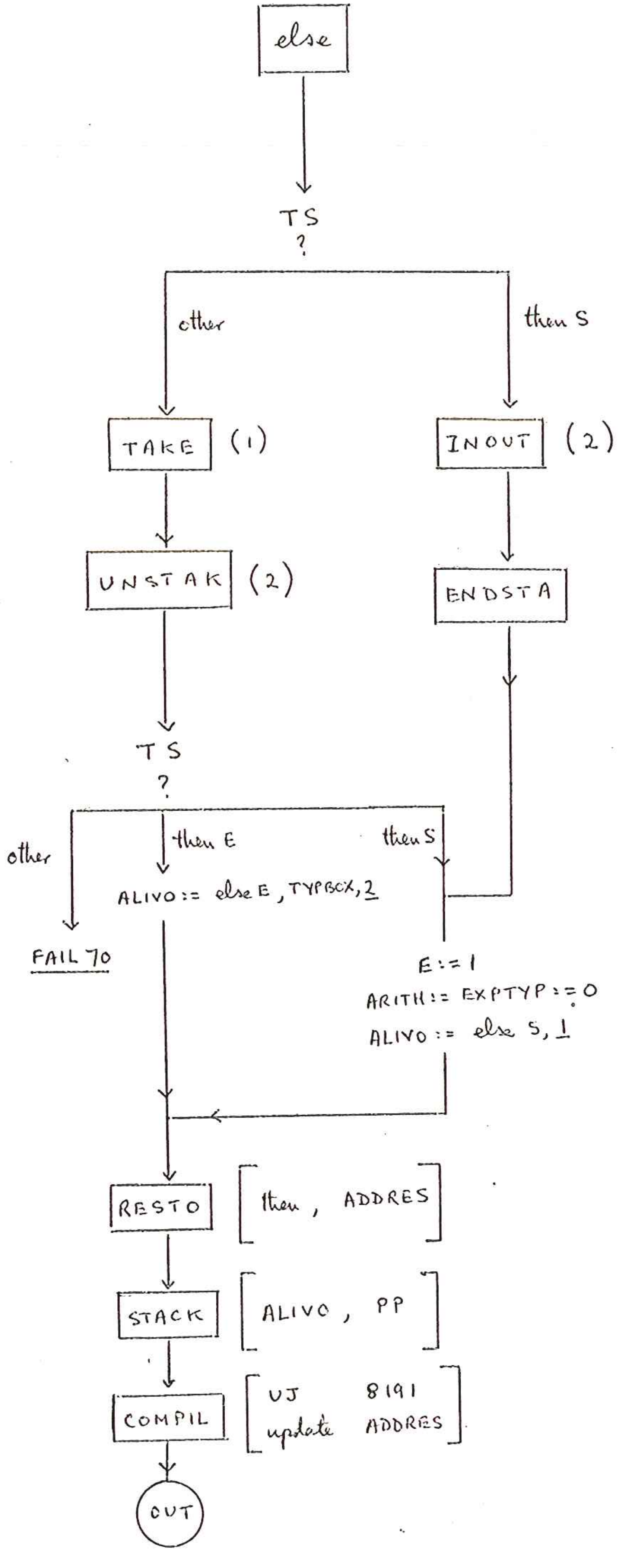


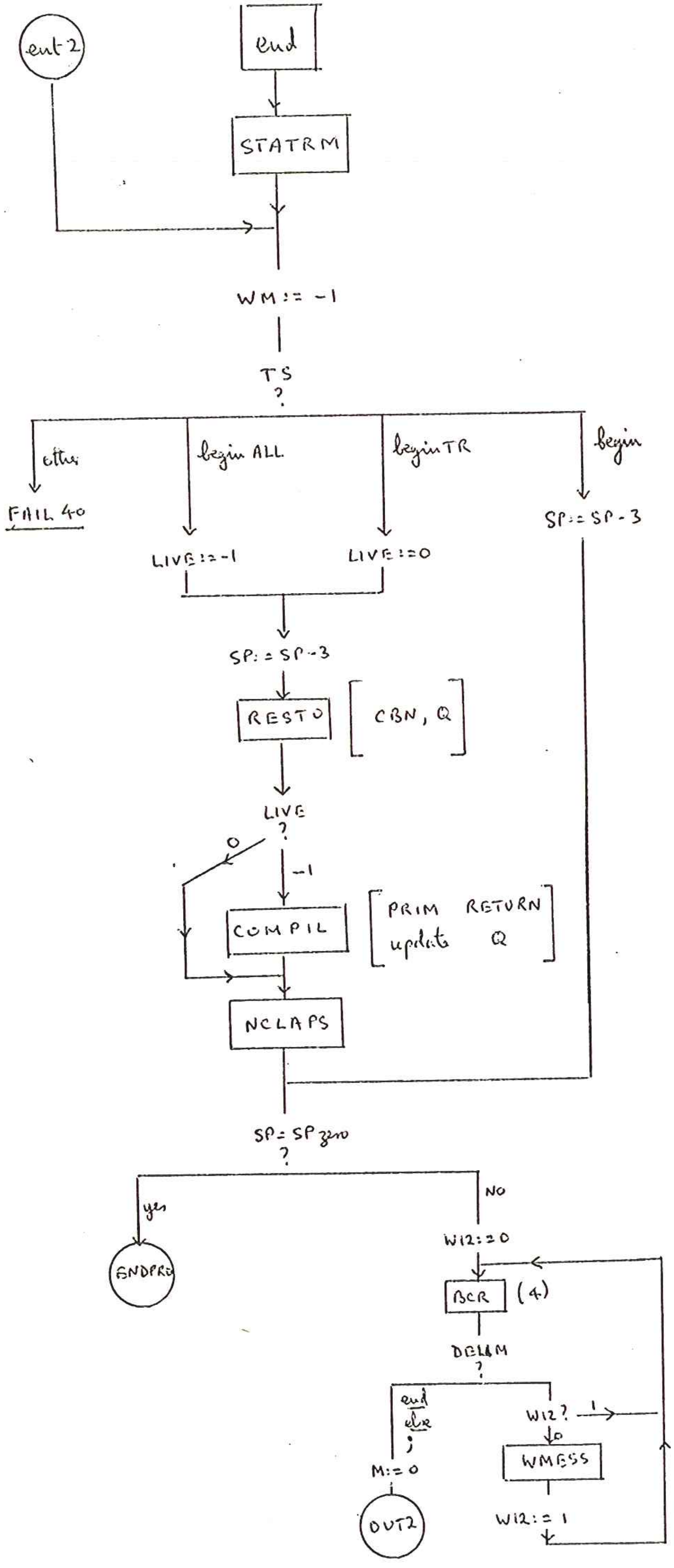
ARENT2 is
an entry
from RSBRAK.


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

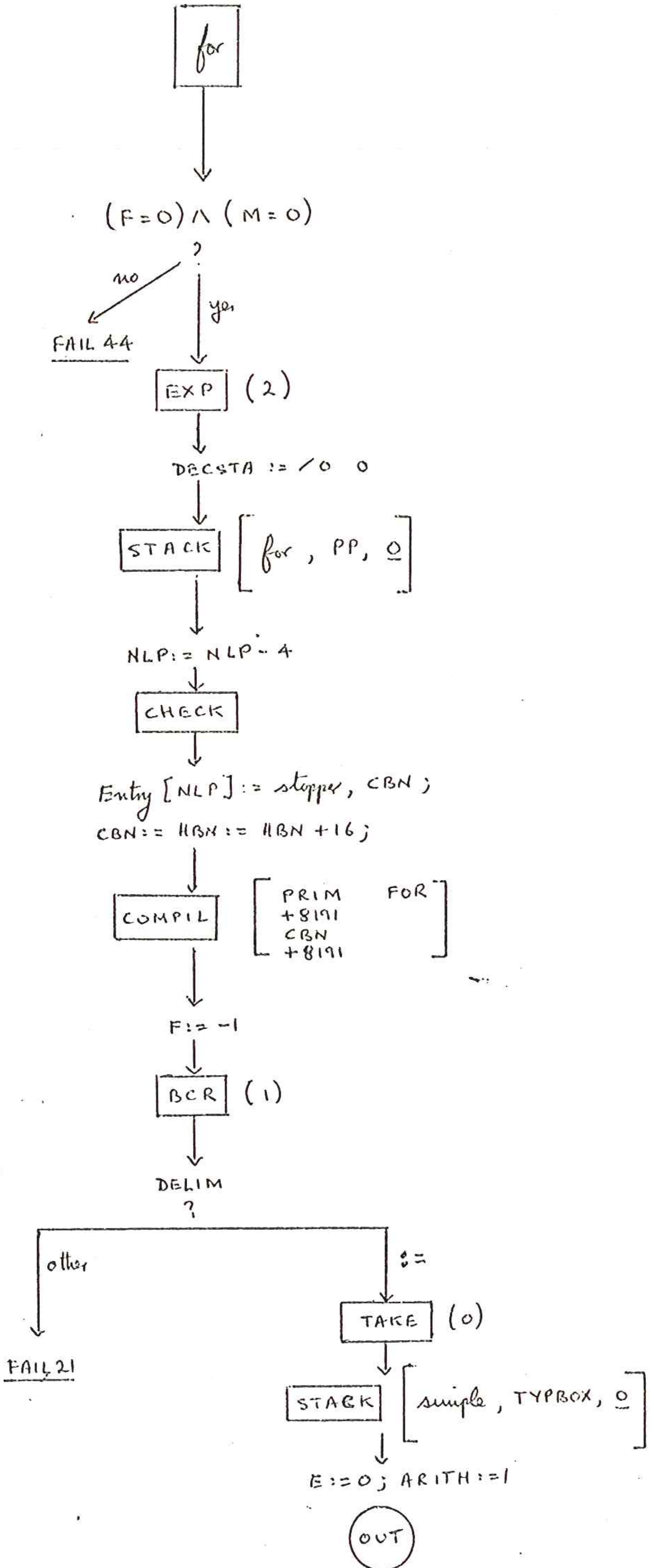

begin
E = 0
?
yes
no
FAIL 60
(F = 1) V (PREC = 1) V (M = 1)
?
yes
no
FAIL 60
DECSTA := 0
TS.
?
ST | STA
other
FAIL 60
STACK
[begin, 0]
OUT


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

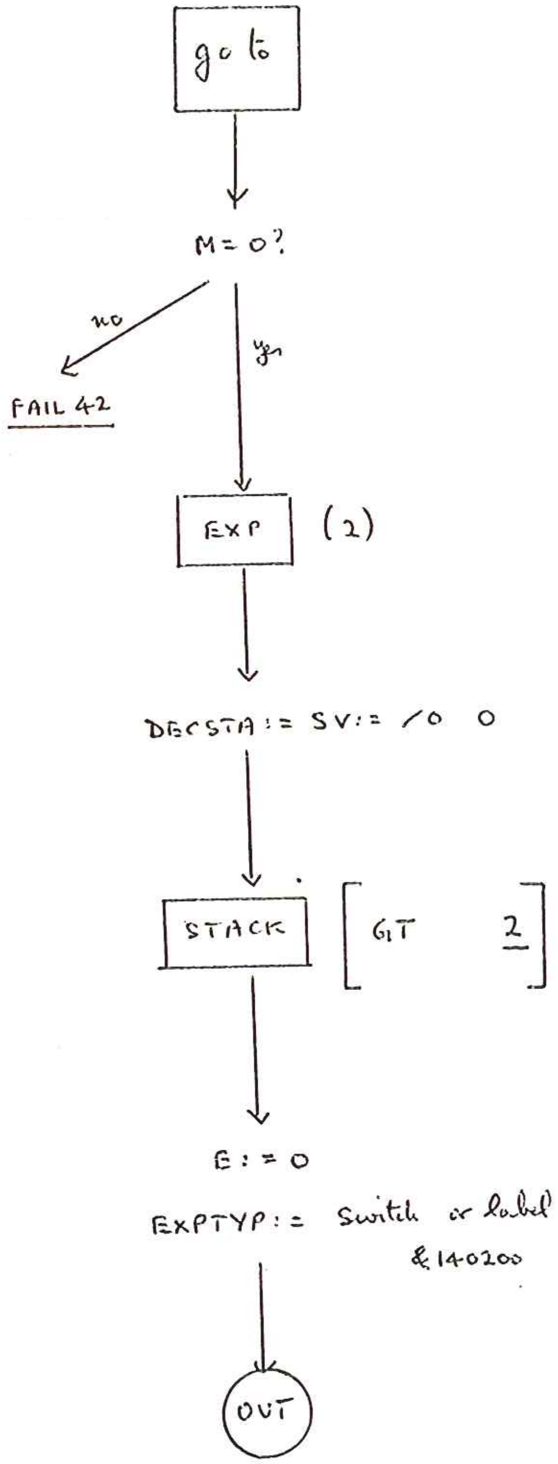

else
TS
?
other
then S
TAKE
(1)
INOUT
(2)
UNSTAK
(2)
ENDSTA
TS
?
other
then E
then S
ALIVO := else E , TYPBOX, 2
FAIL 70
E := 1
ARITH := EXPTYP := 0
ALIVO := else S, 1
RESTO
then , ADDRES
STACK
ALIVO , PP
COMPIL
UJ 8191
update ADDRES
OUT

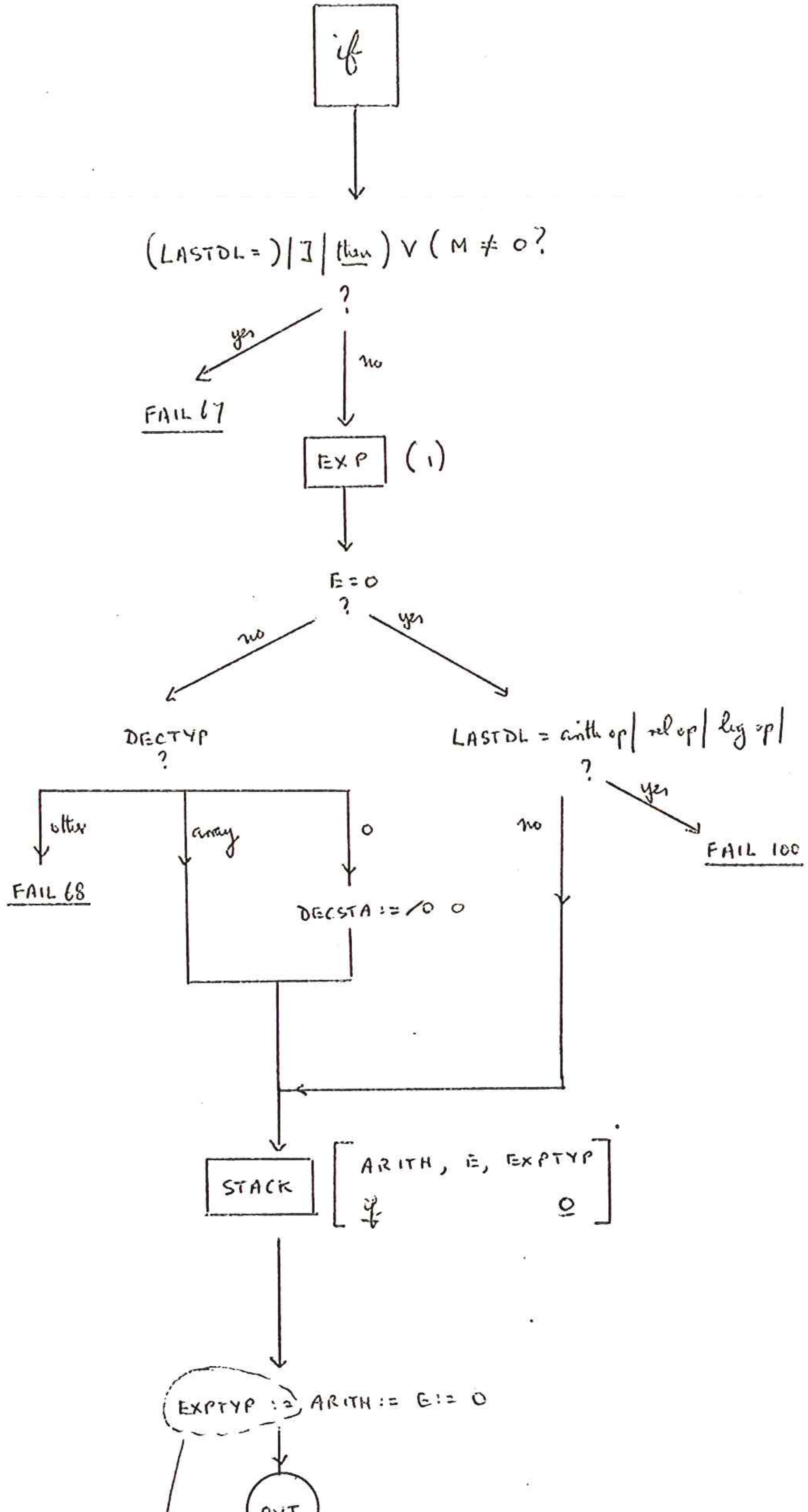


ENT2 is an entry from FAIL


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

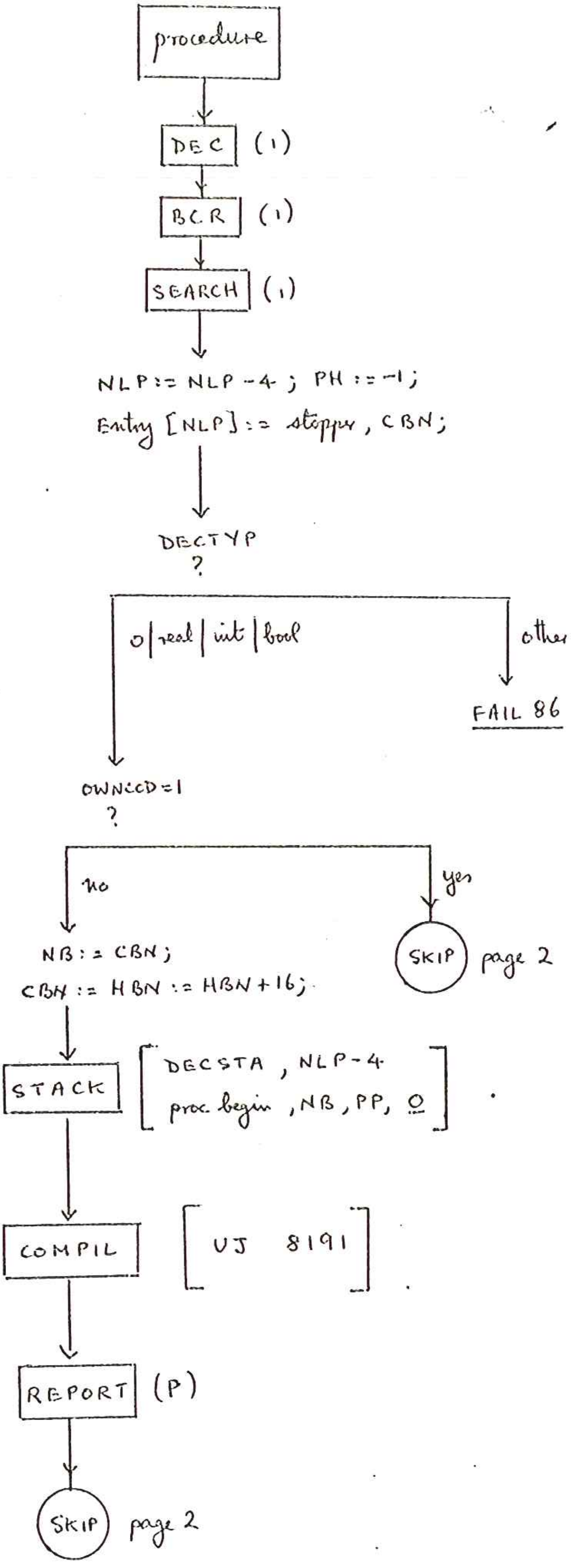
stopper is -1


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



May76 do not set EXPTYP=0!

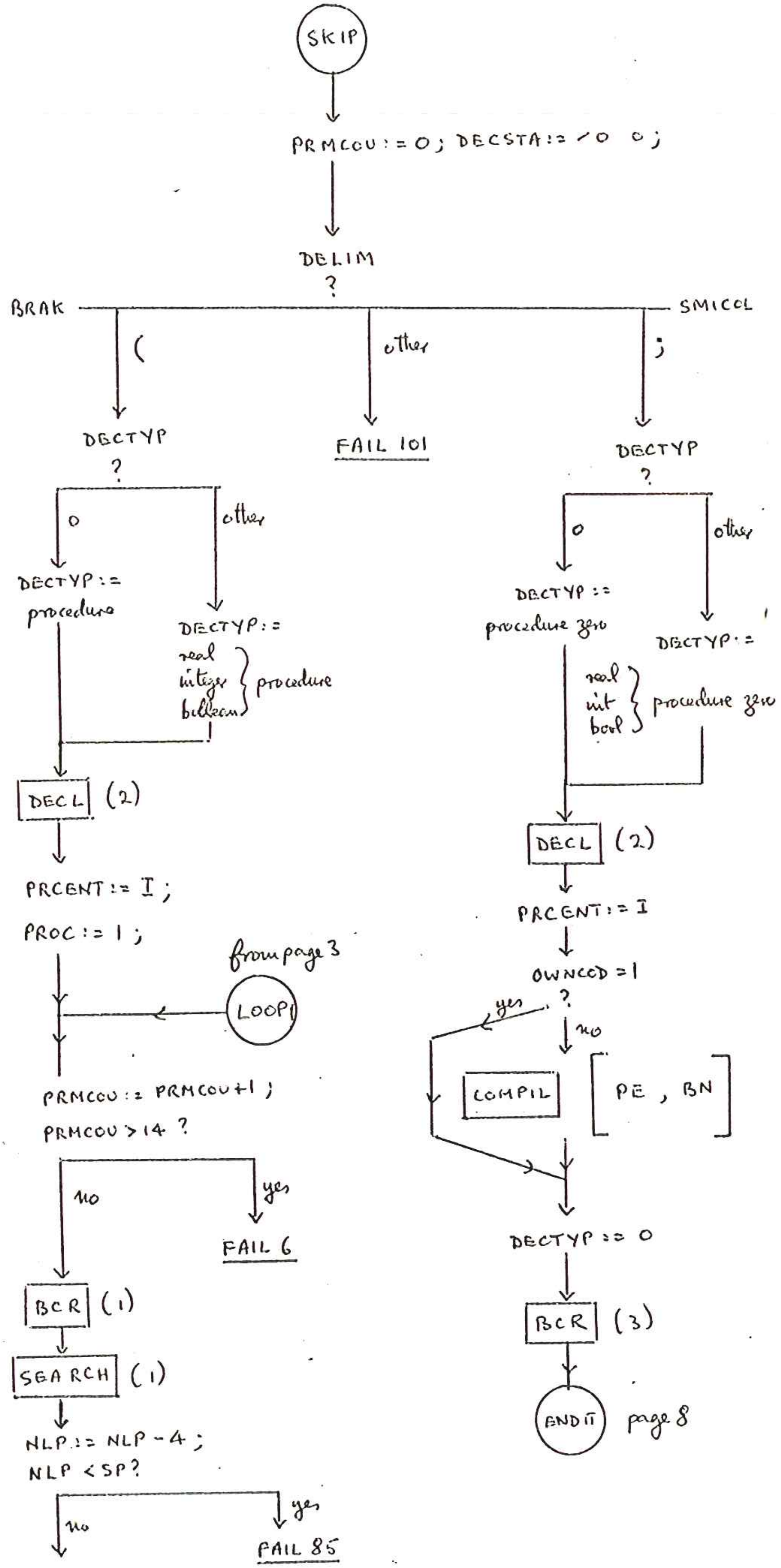
page 1 of 8



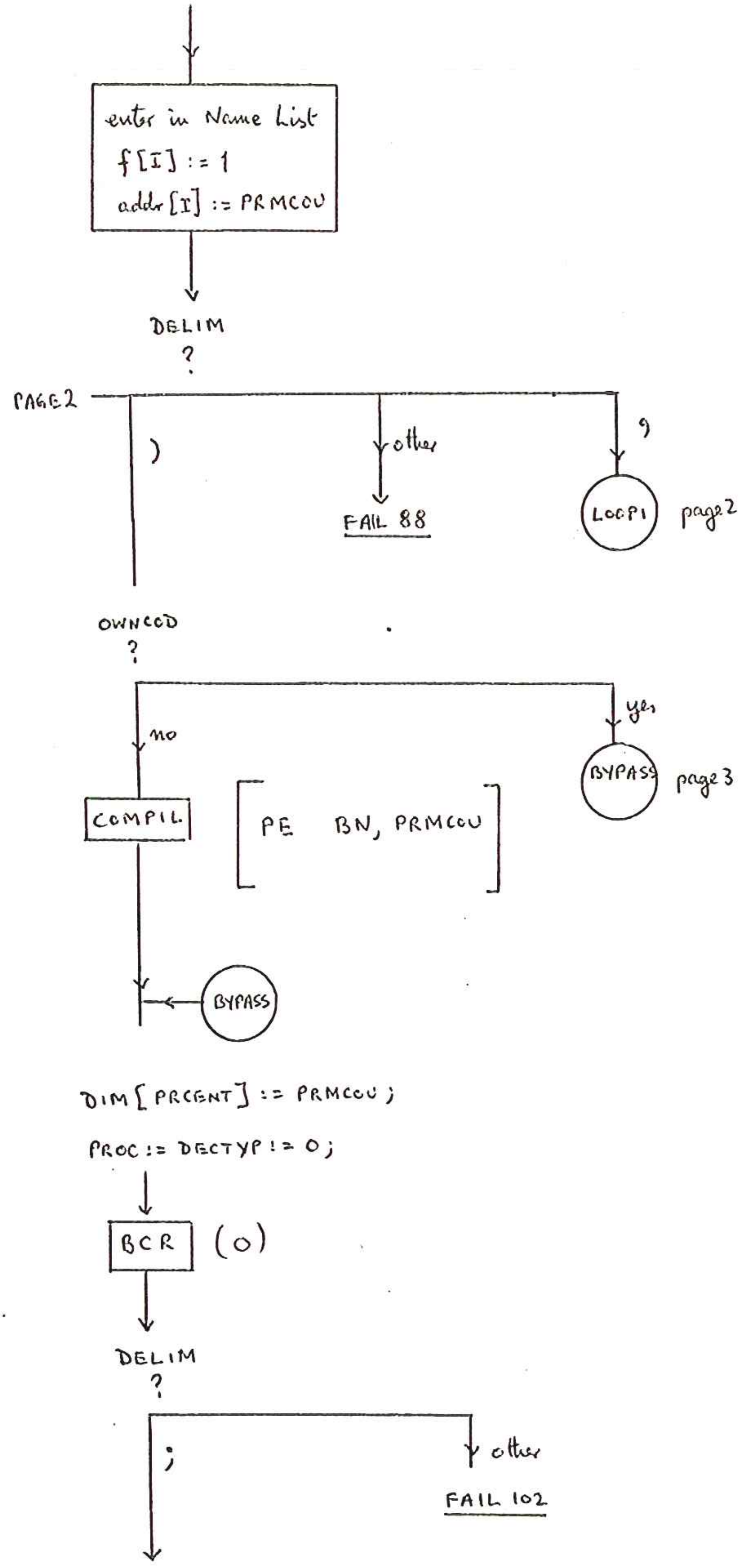
PH is procedure heading marker for FAIL

report procedure name if in report mode

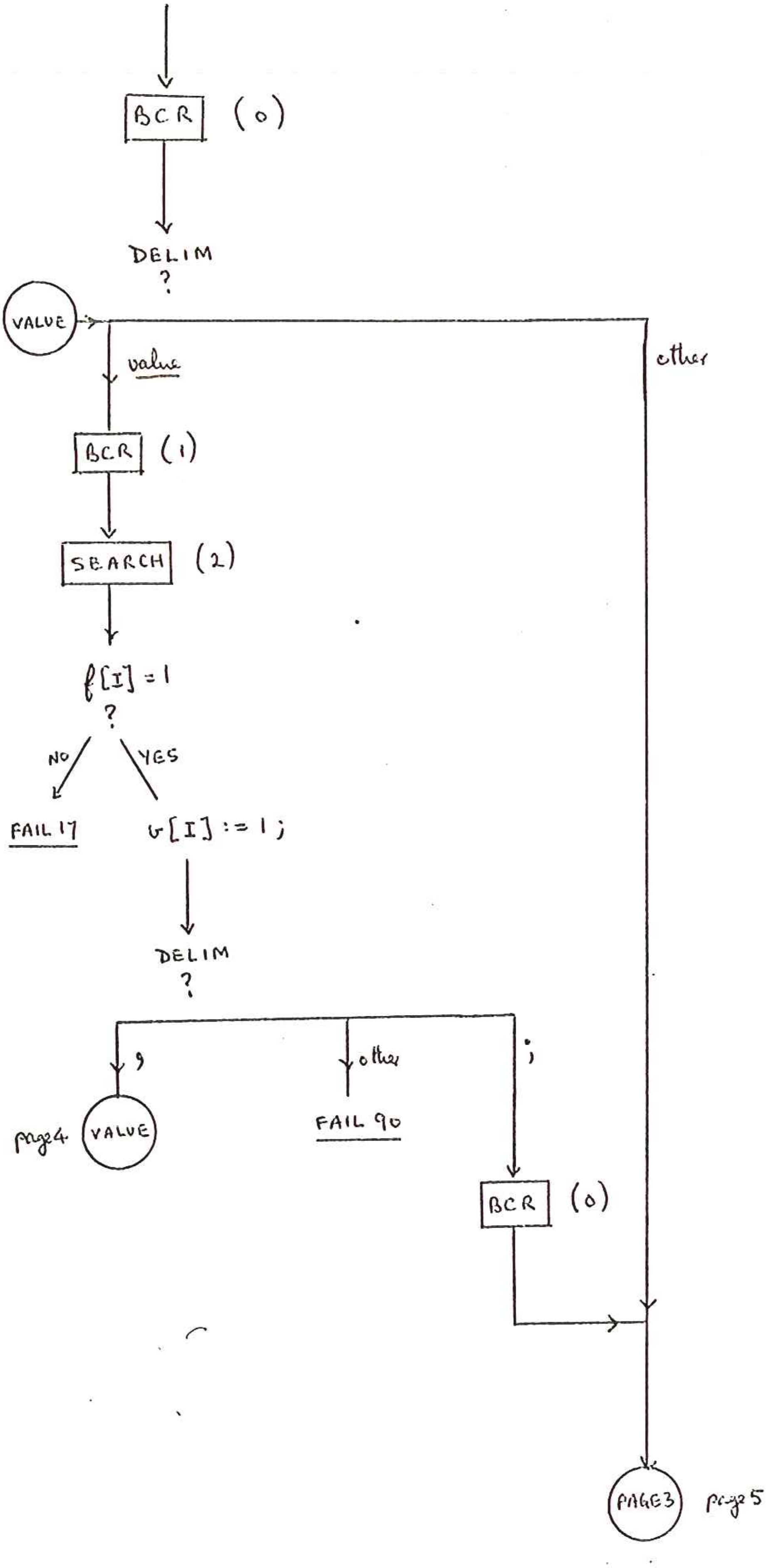
procedure (continued)



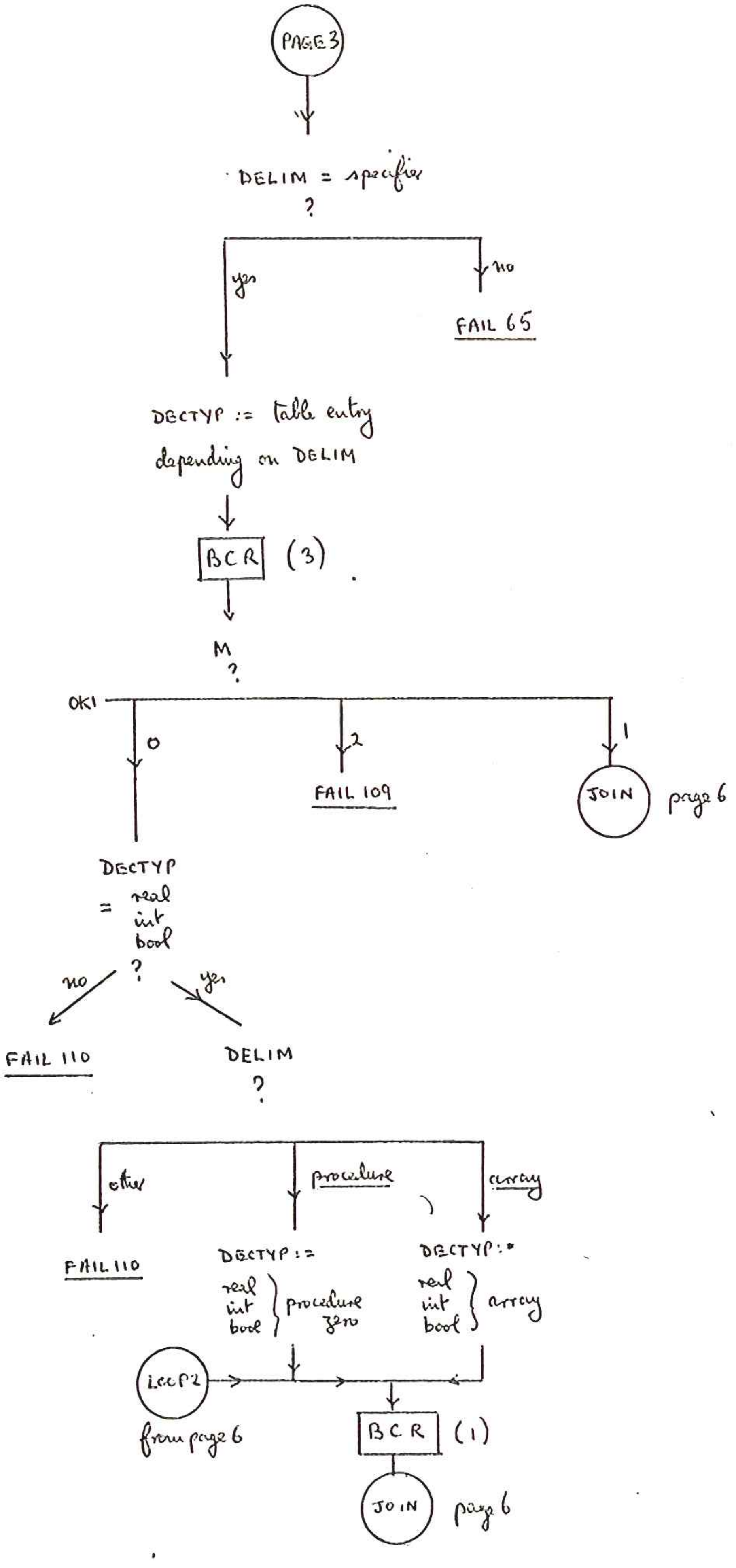
procedure (cont'd)



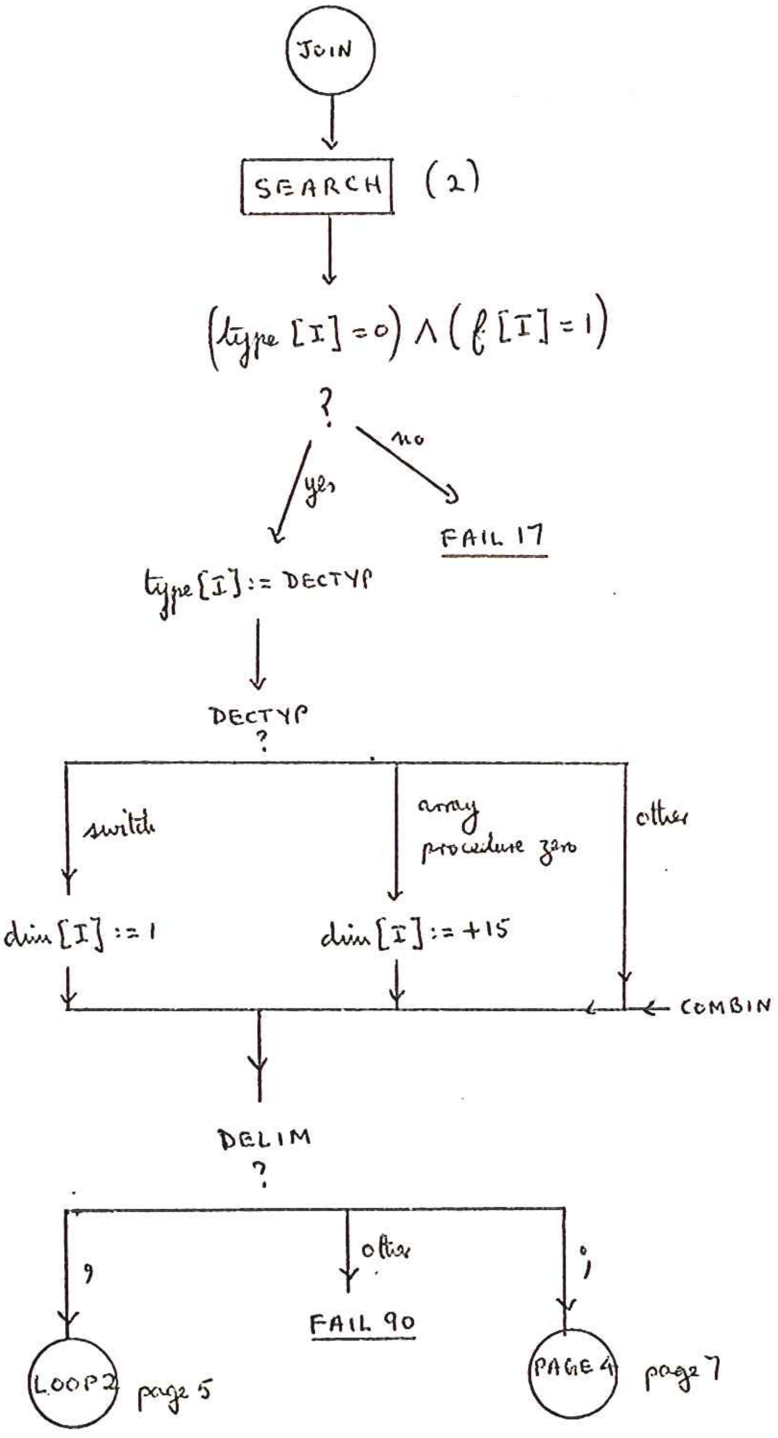
procedure (cont'd)



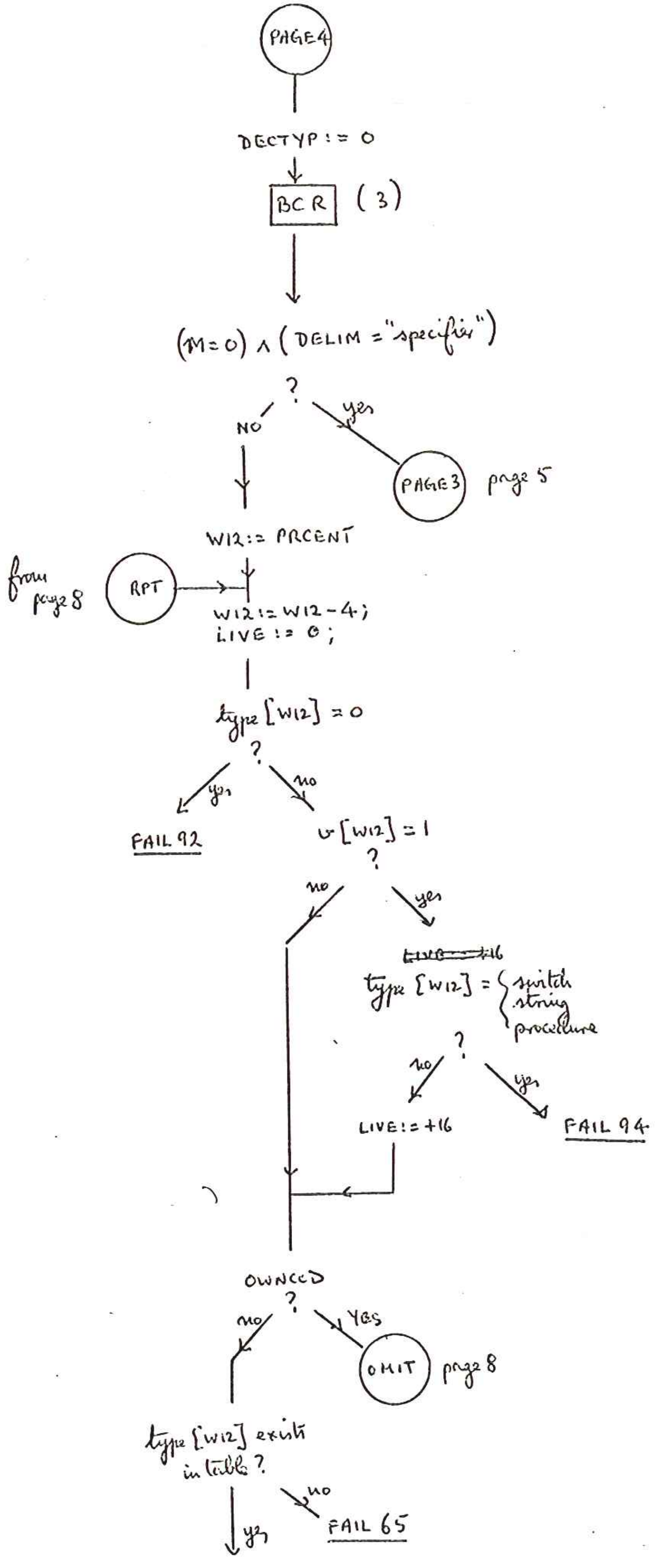
procedure (cont'd)



procedure (cont'd)

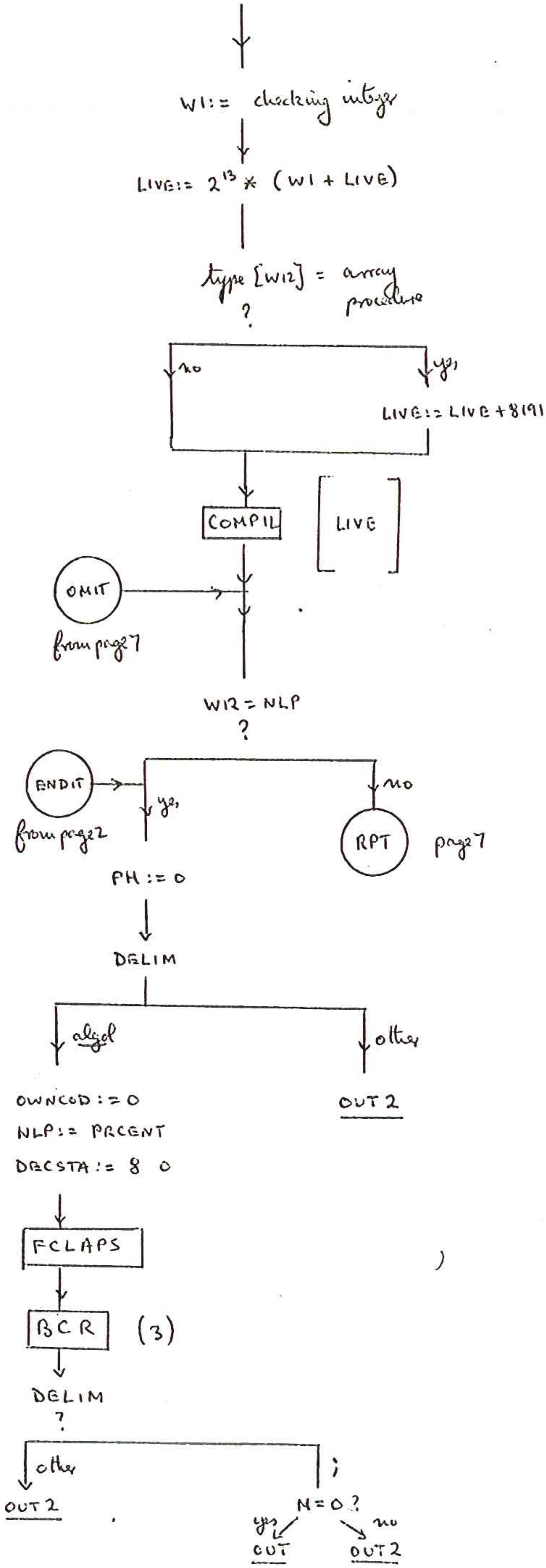


procedure (cont'd)

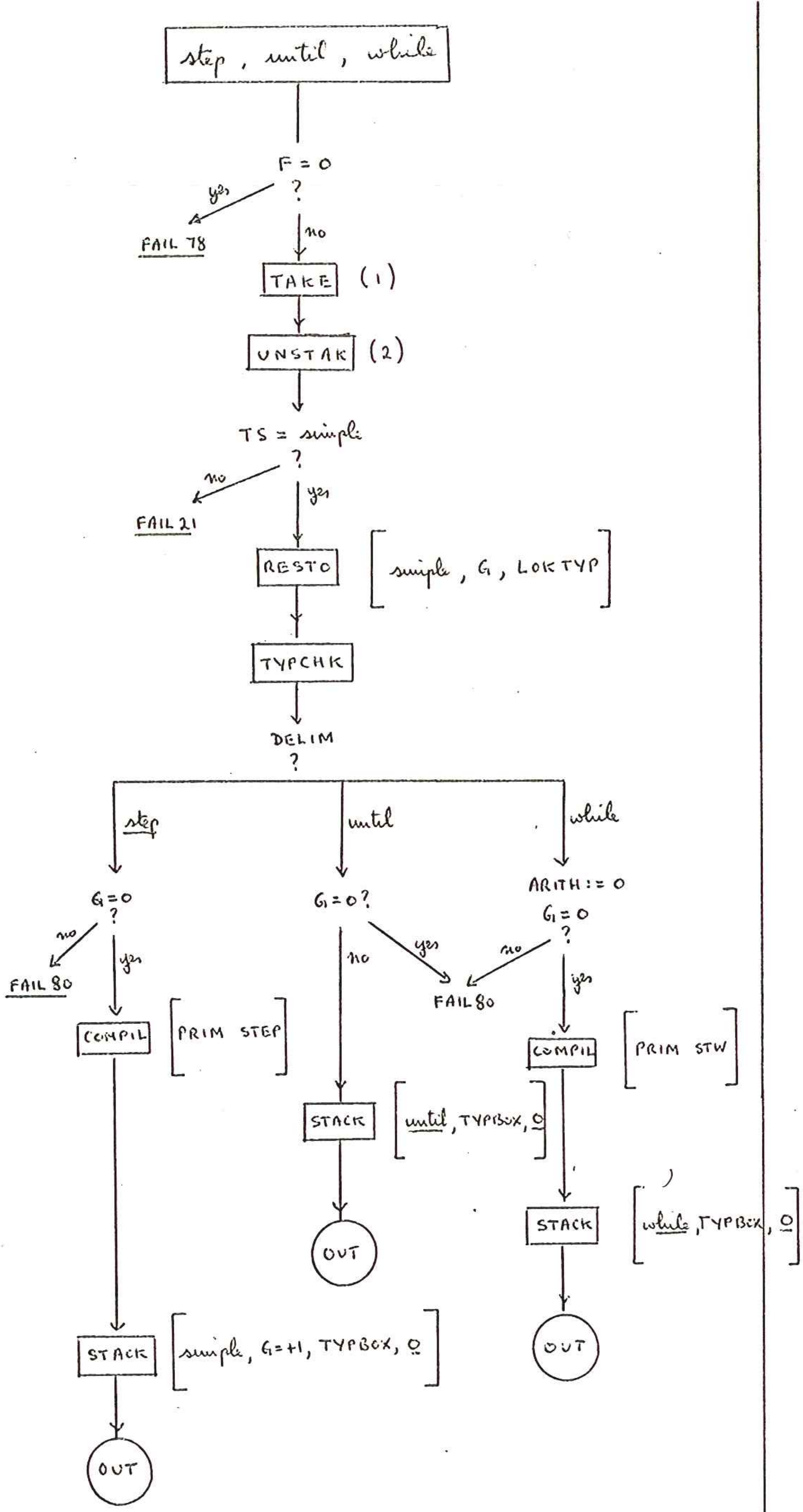


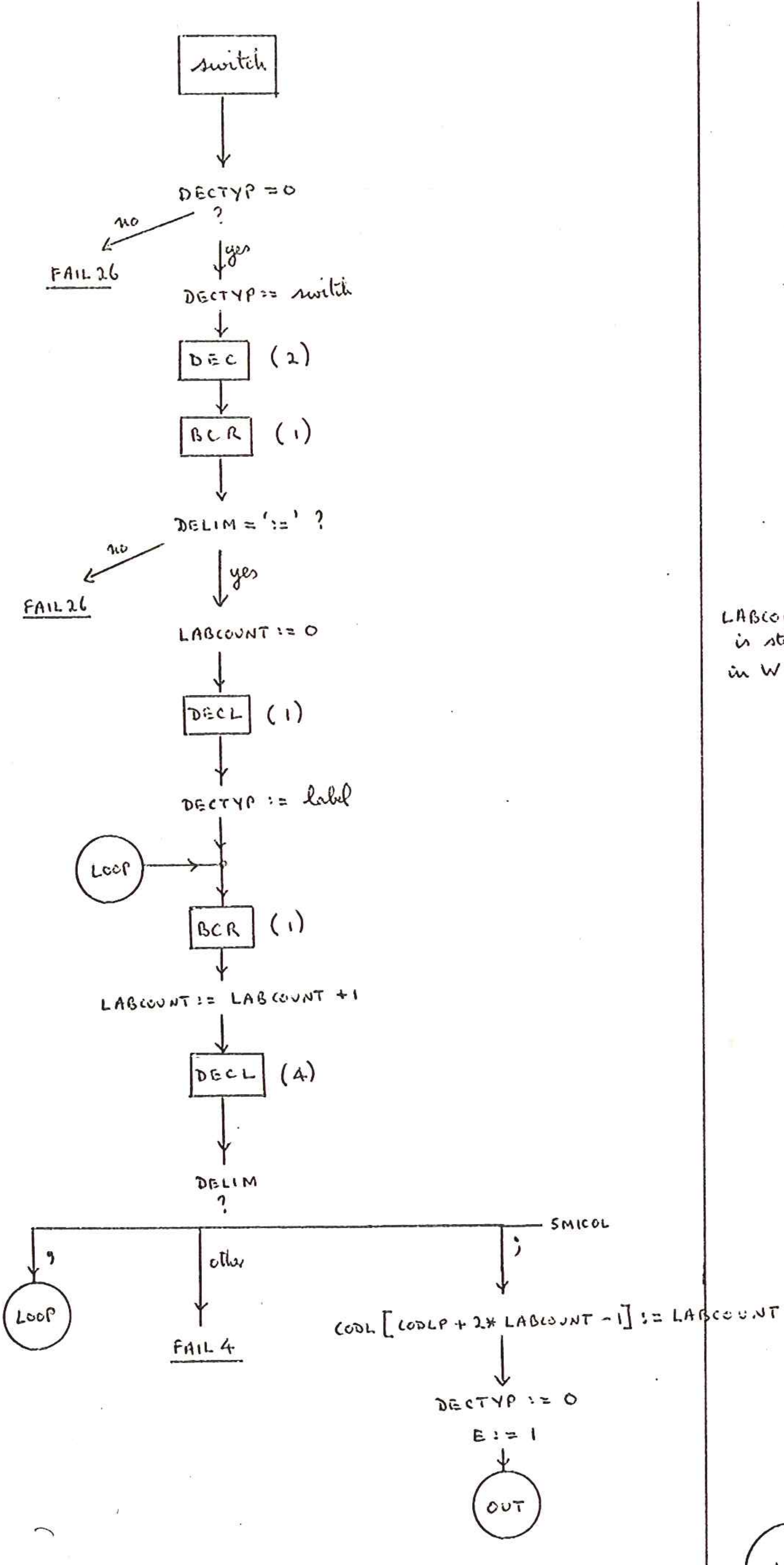
All procedures will be procedure zero at this stage

procedure (cont'd)

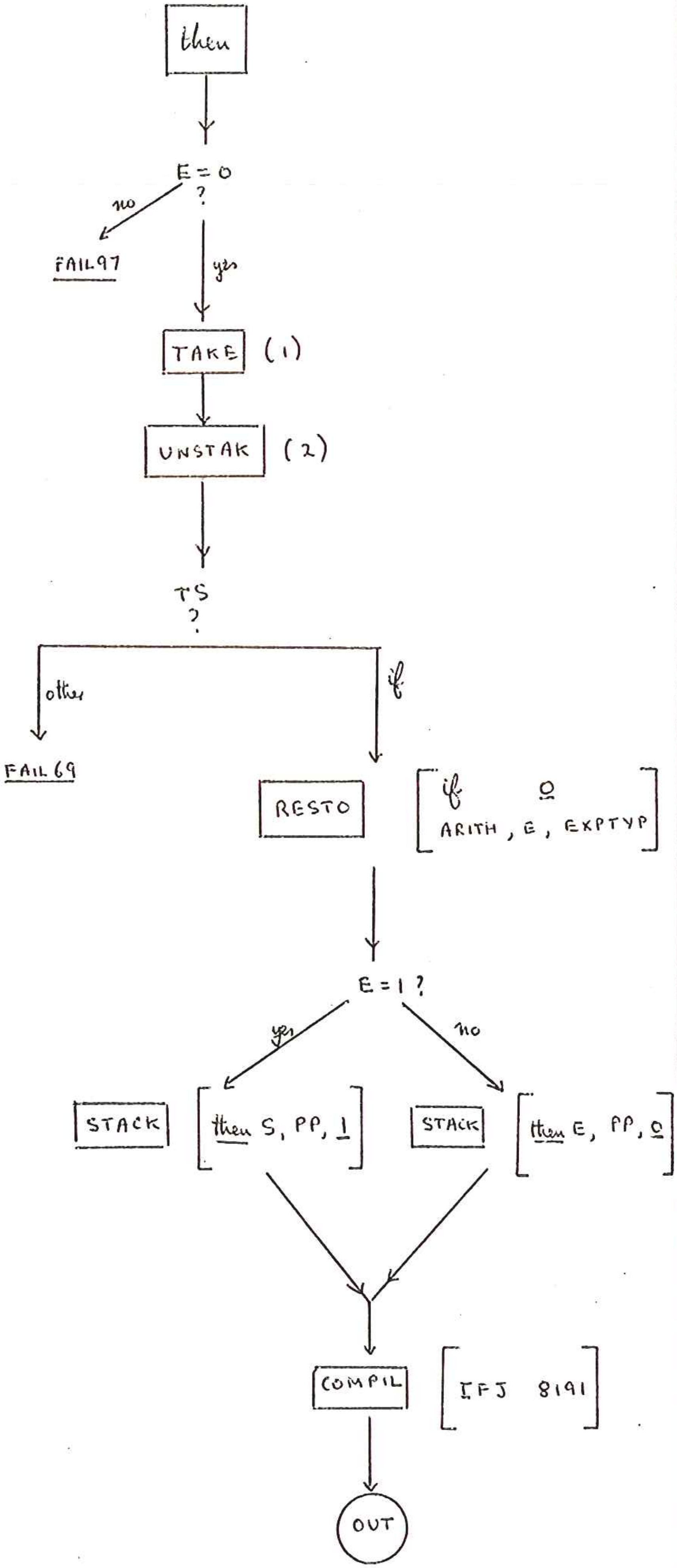


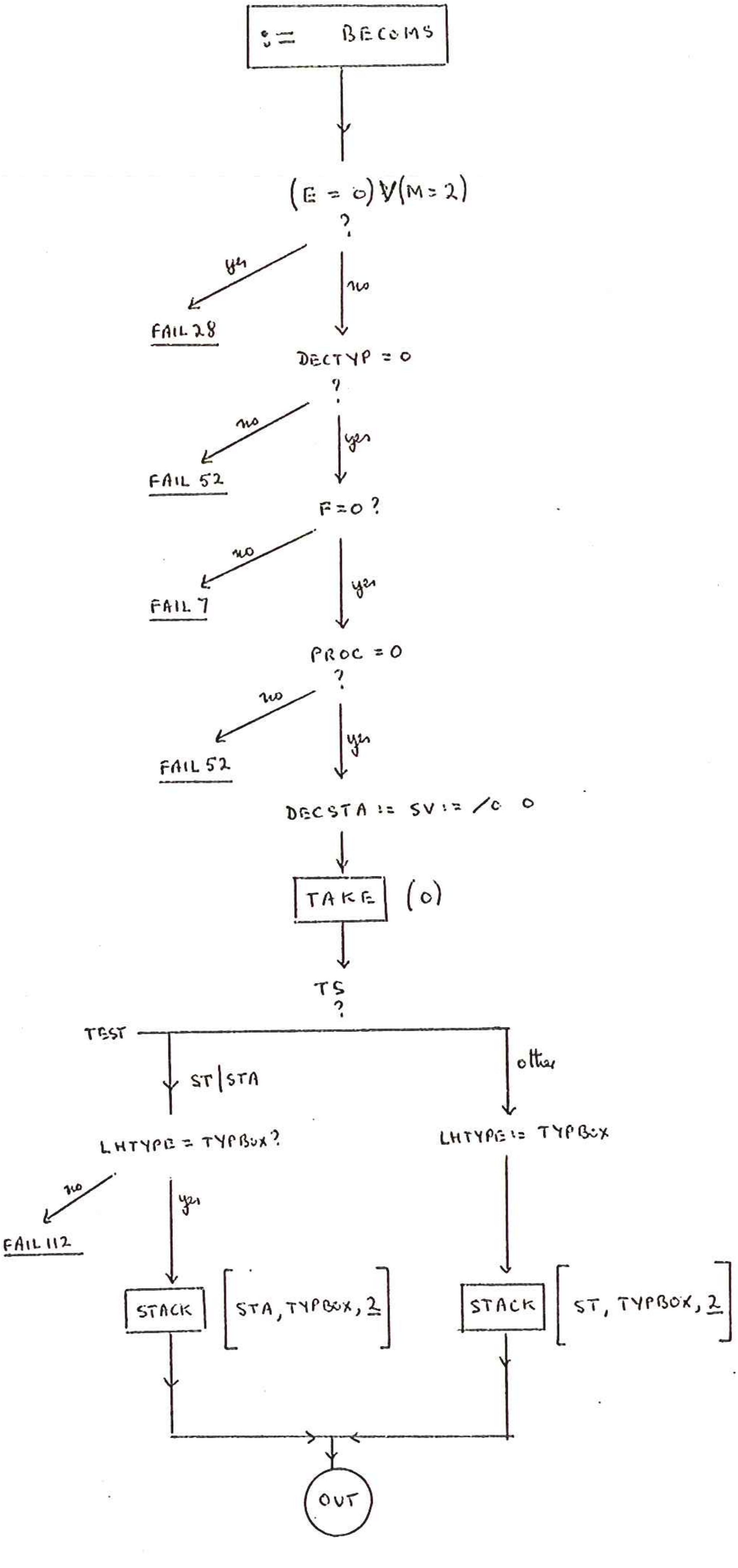
LIVE contains a parameter checking word see pard manual


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
G = 0 ?
no
yes
FAIL 80
STACK
until, TYPBOX, 0
OUT
ARITH := 0
G = 0 ?
no
yes
COMPIL
PRIM STW
STACK
while, TYPBOX, 0
OUT

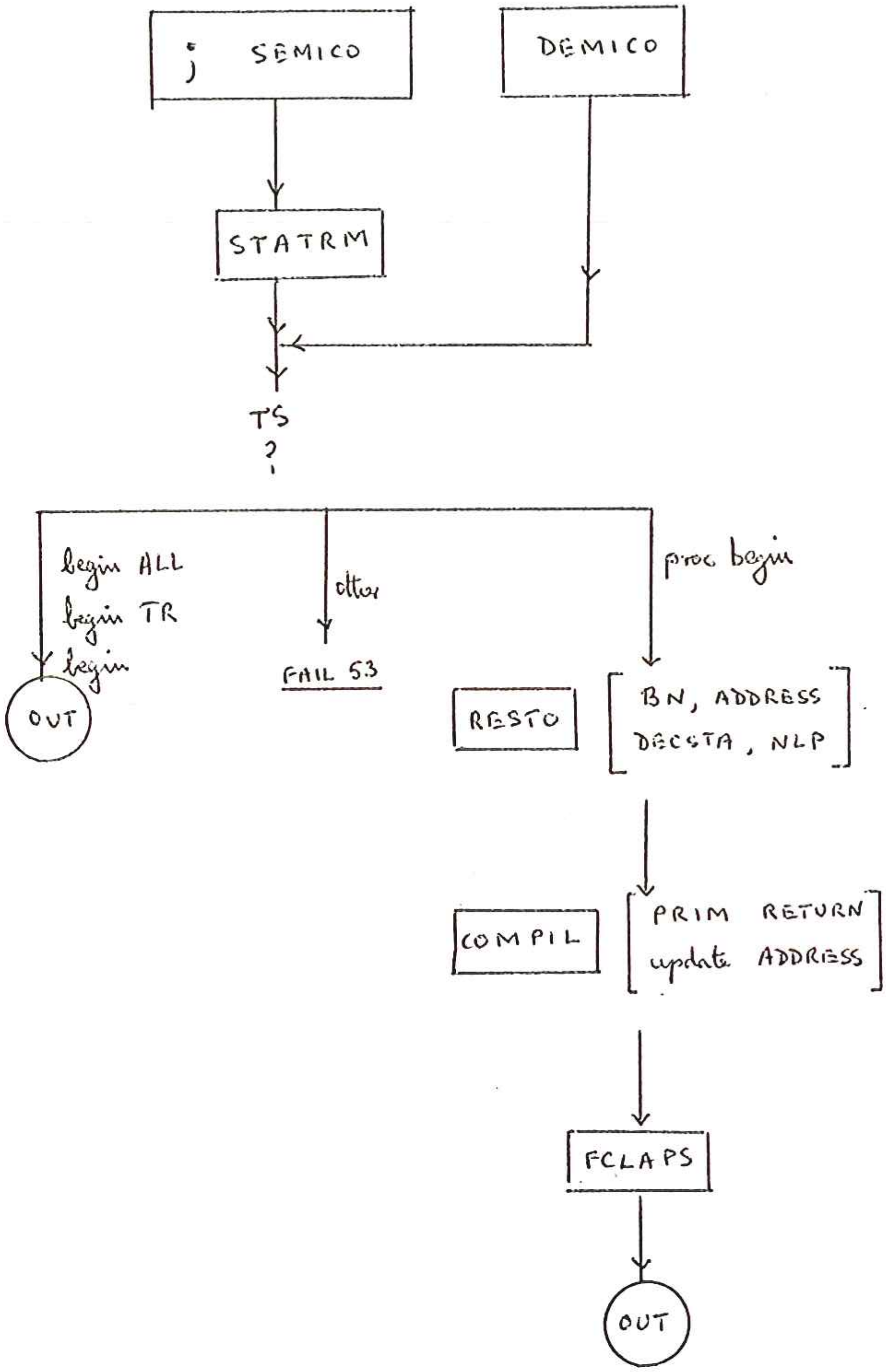


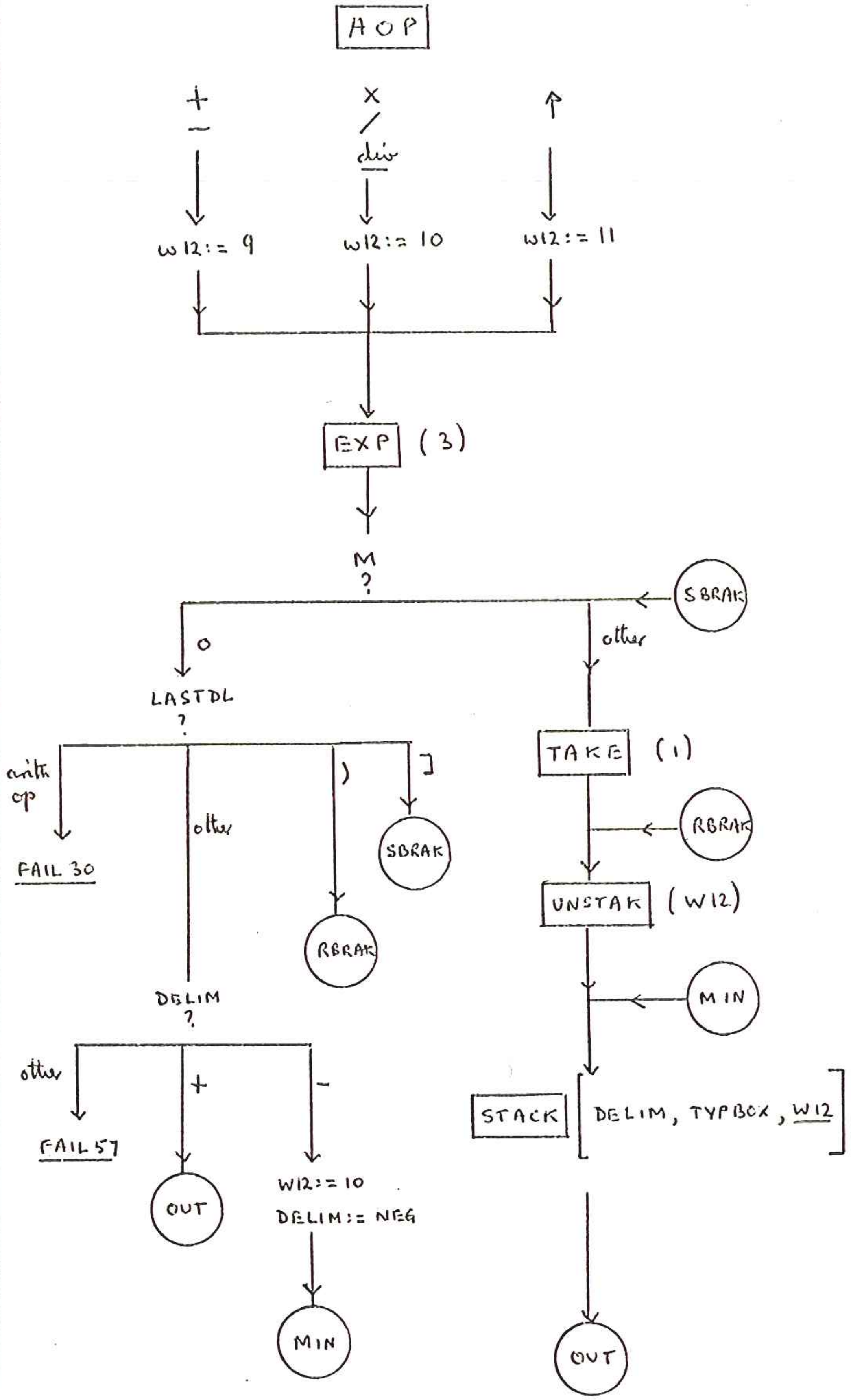
LABCOUNT is stored in W+12


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 variables


; SEMICO
DEMICO
STATRM
TS
?
begin ALL
begin TR
begin
other
proc begin
OUT
FAIL 53
RESTO
BN, ADDRESS
DECSTA, NLP
COMPIL
PRIM RETURN
update ADDRESS
FCLAPS
OUT

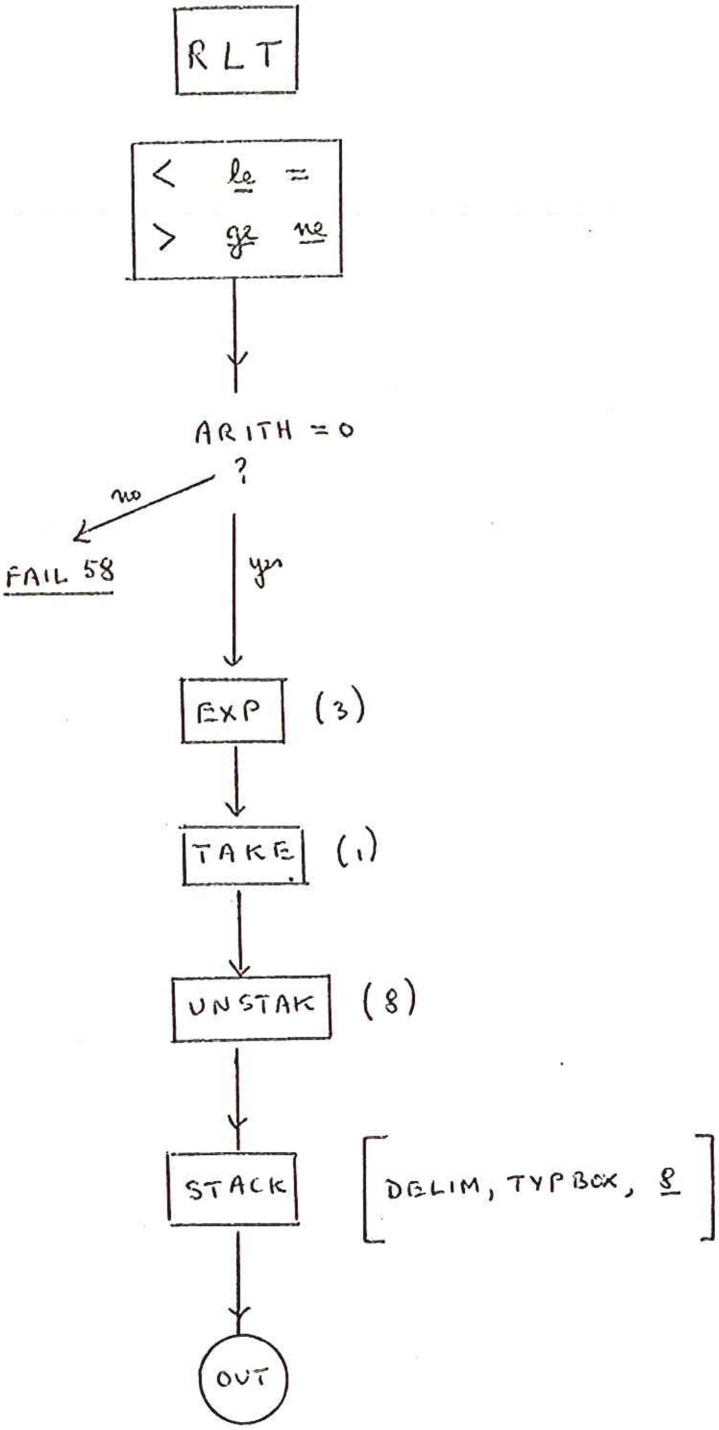


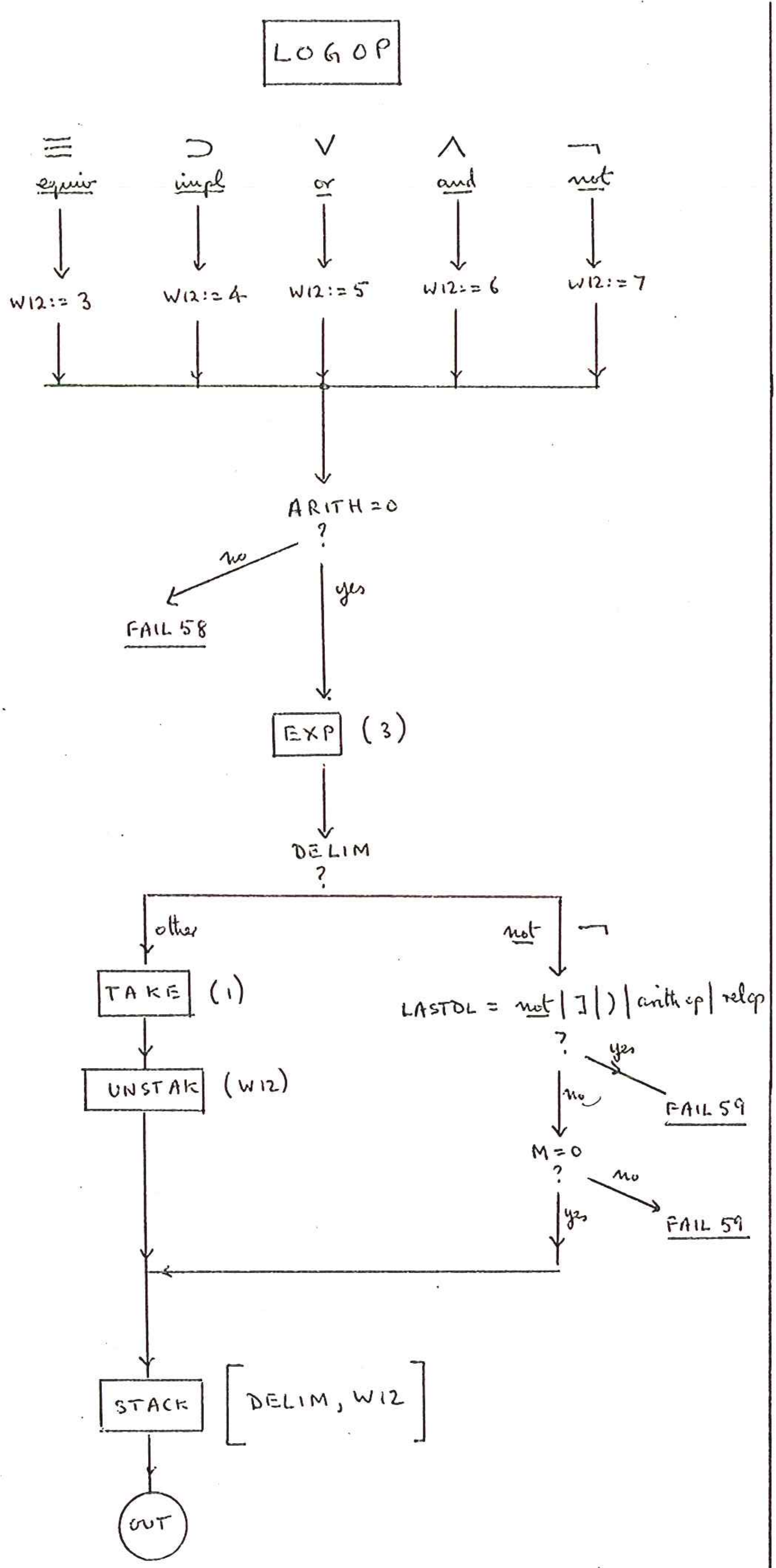
w12 = stack priority

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

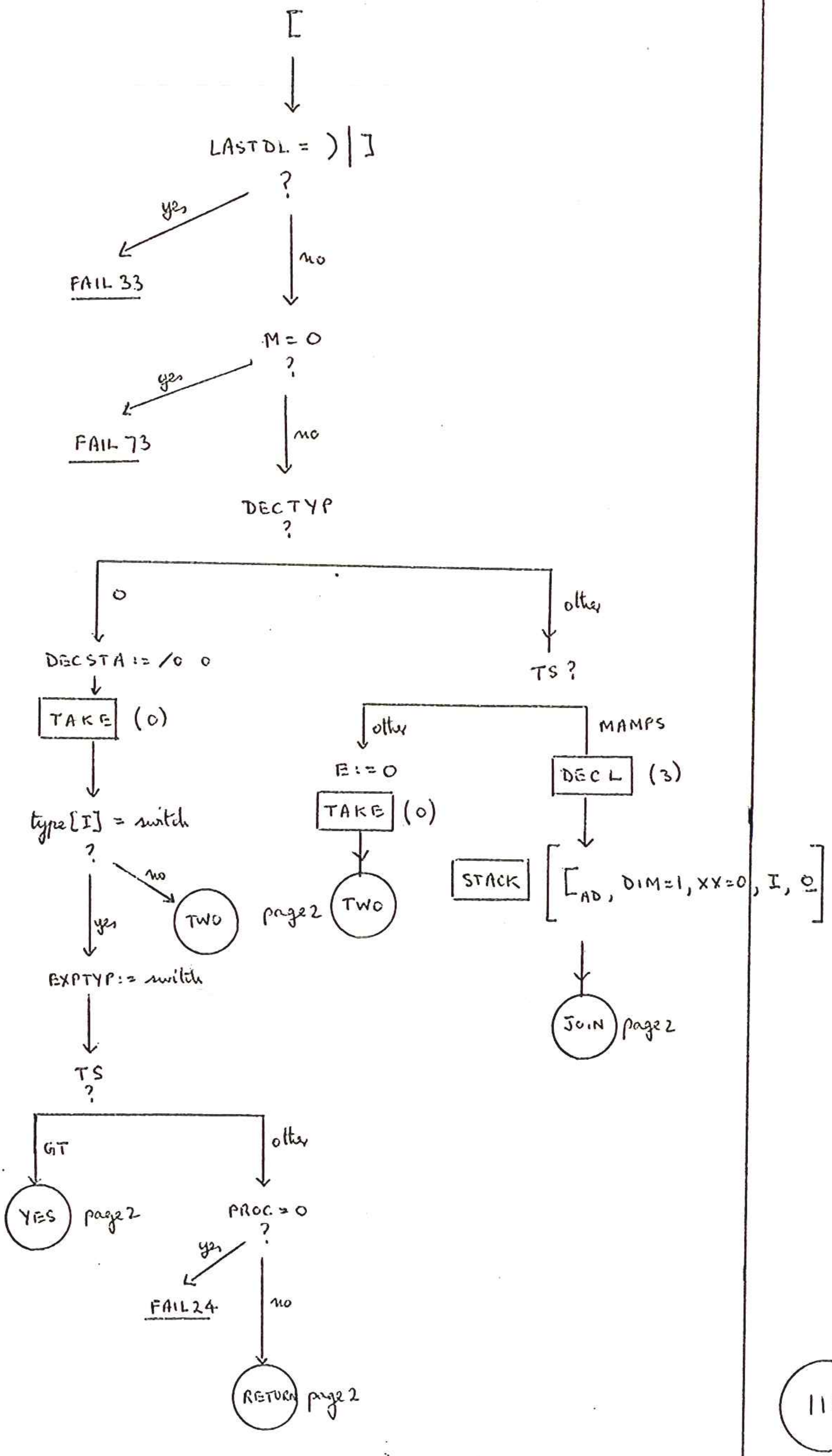
unary + is ignored

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

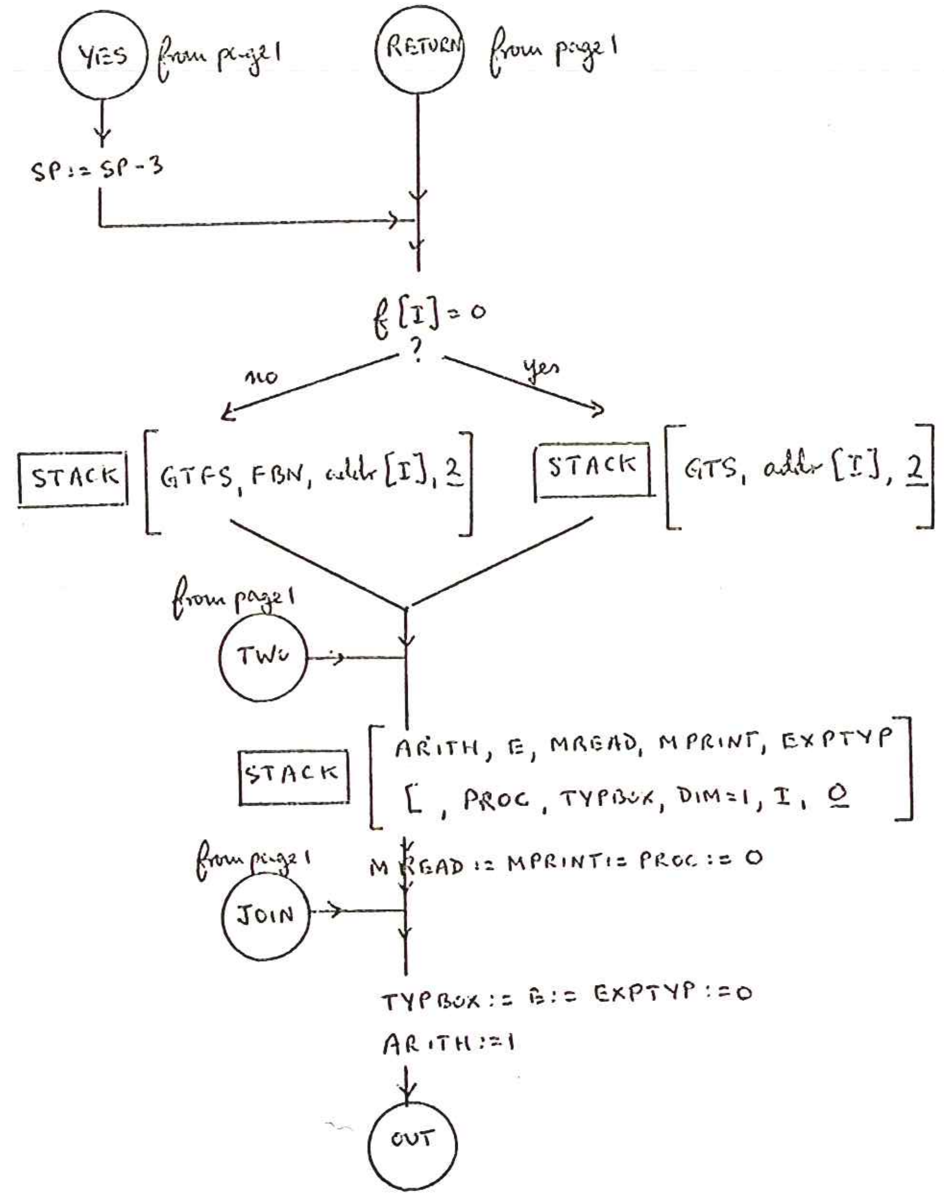

LOGOP
≡
⊃
∨
∧
¬
equiv
impl
or
and
not
W12:= 3
W12:= 4
W12:= 5
W12:= 6
W12:= 7
ARITH = 0
?
no
yes
FAIL 58
EXP
(3)
DELIM
?
other
not
¬
TAKE
(1)
UNSTAK
(W12)
LASTDL = not |] |) | arith op | relop
?
yes
no
FAIL 59
M = 0
?
no
yes
FAIL 59
STACK
[DELIM, W12]
OUT

LSBRAK

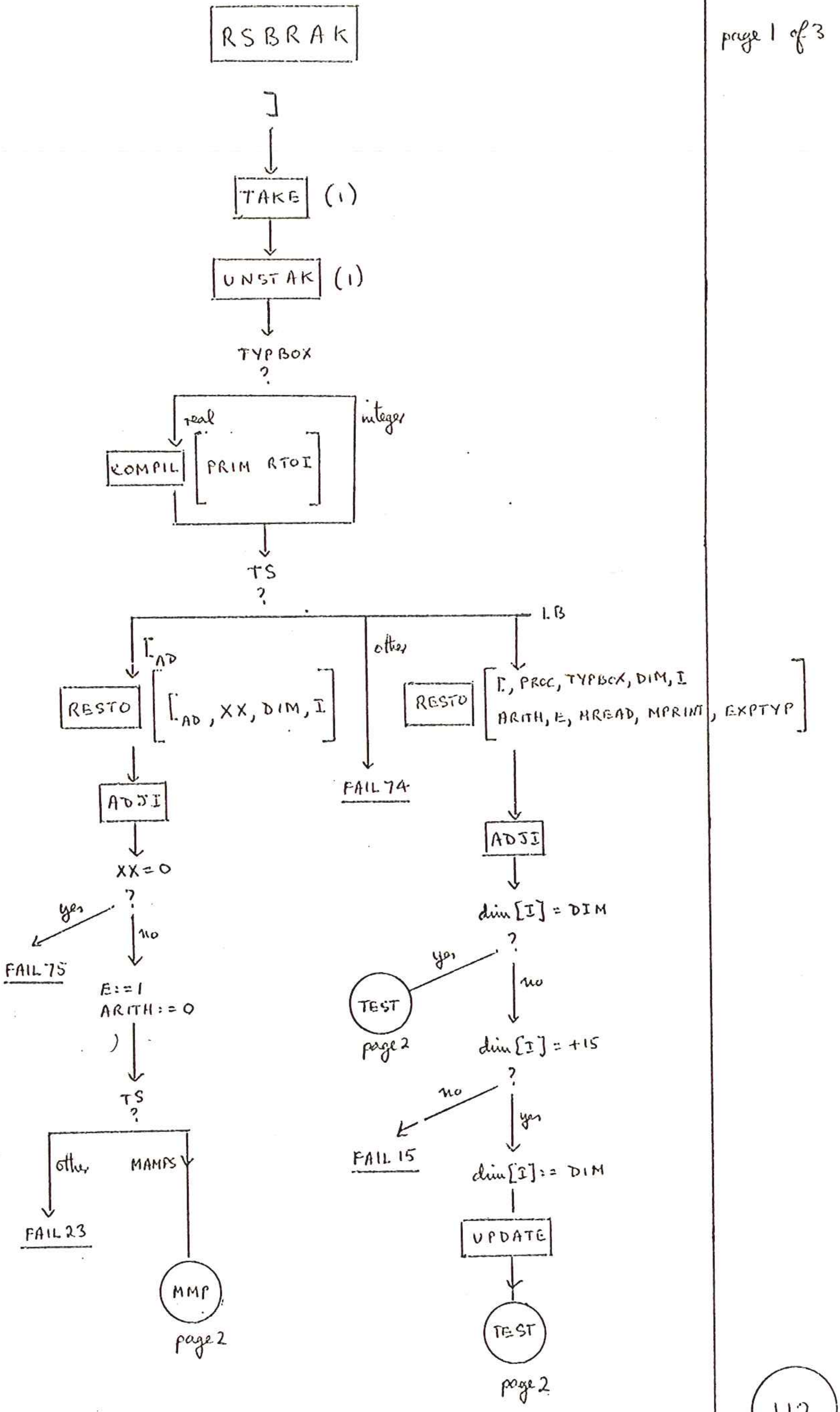

[
LASTDL =) |]
?
yes
FAIL 33
no
M = 0
?
yes
FAIL 73
no
DECTYP
?
0
other
DECSTA := /0 0
TAKE
(0)
type[I] = switch
?
no
TWO
yes
EXPTYP := switch
TS
?
GT
YES
page 2
other
PROC = 0
?
yes
FAIL 24
no
RETURN
page 2
TS ?
other
E := 0
TAKE
(0)
page 2
TWO
MAMPS
DECL
(3)
STACK
[[AD, DIM=1, XX=0, I, 0]
JOIN
page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

LSBRAK contd

page 2 of 2

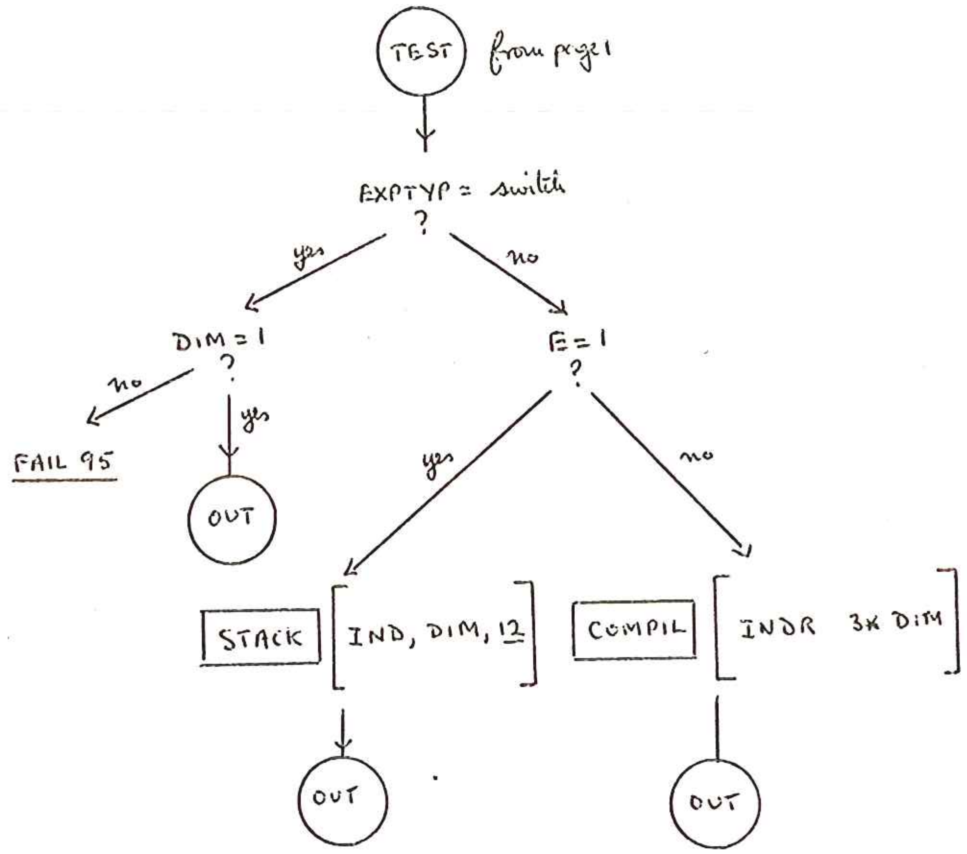


page 1 of 3

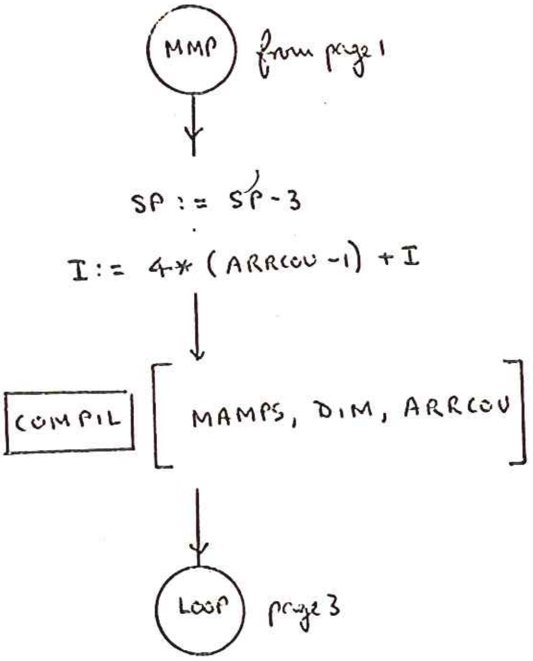


RSBRAK
]
TAKE (1)
UNSTAK (1)
TYPBOX ?
real
integer
KOMPIL
PRIM RTOI
TS ?
[AD
other
I.B
RESTO
[AD , XX, DIM, I
FAIL 74
RESTO
[, PROC, TYPBOX, DIM, I
ARITH, E, MREAD, MPRINT, EXPTYP
ADJI
XX = 0 ?
yes
no
FAIL 75
E := 1
ARITH := 0
)
TS ?
other
MAMPS
FAIL 23
MMP
page 2
ADJI
dim[I] = DIM ?
yes
no
TEST
page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

RSBRAK cont'd

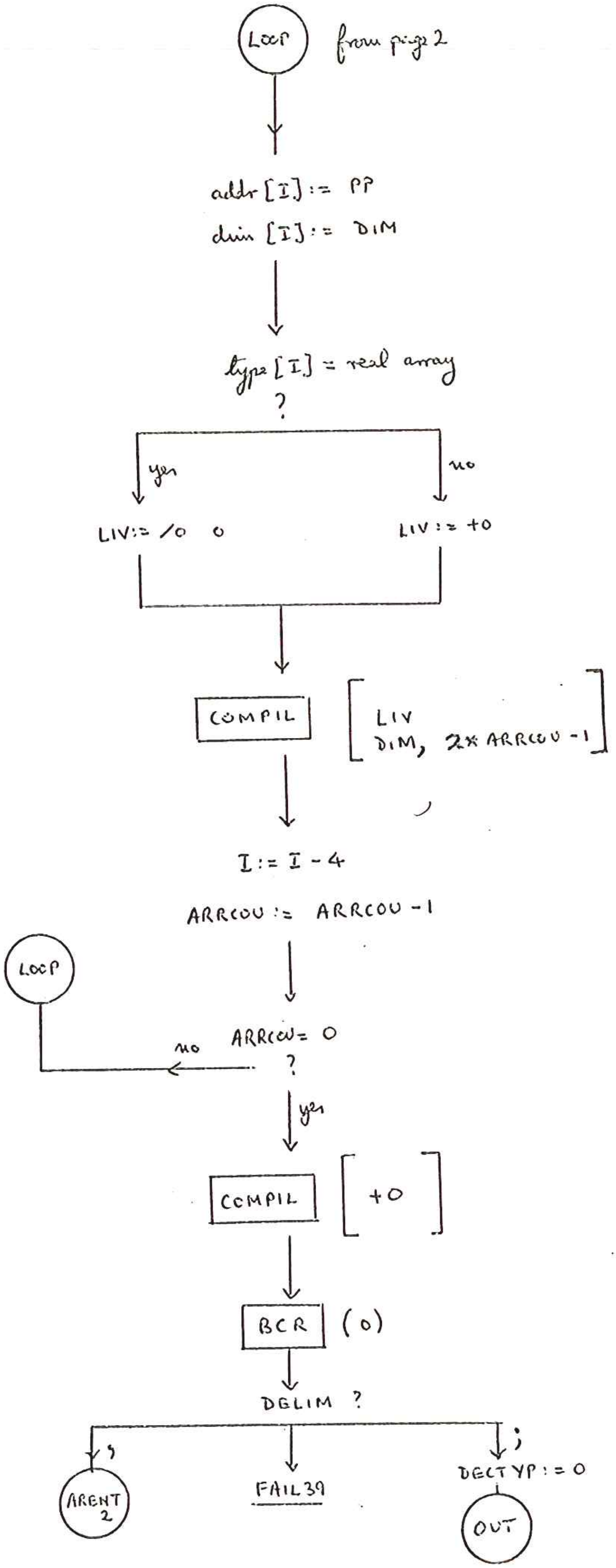


DIM = dim [I]



RSBRAK contd

page 3 of 3



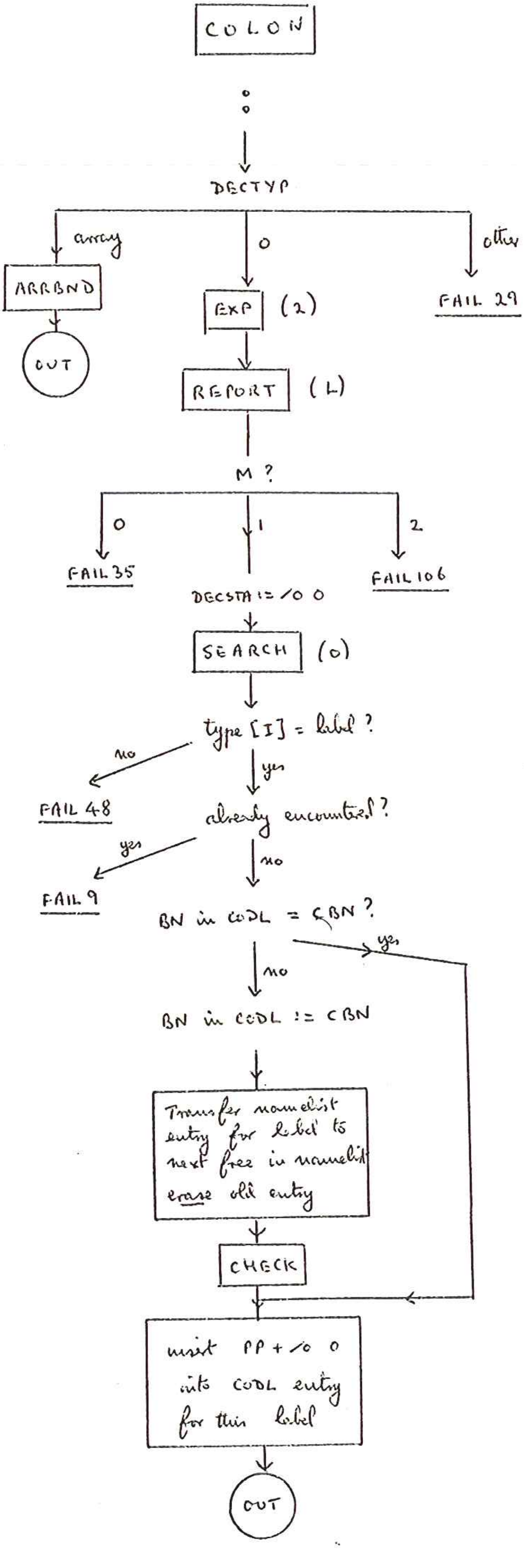
enter details in namelist

compile array pair

I is reset to next array name

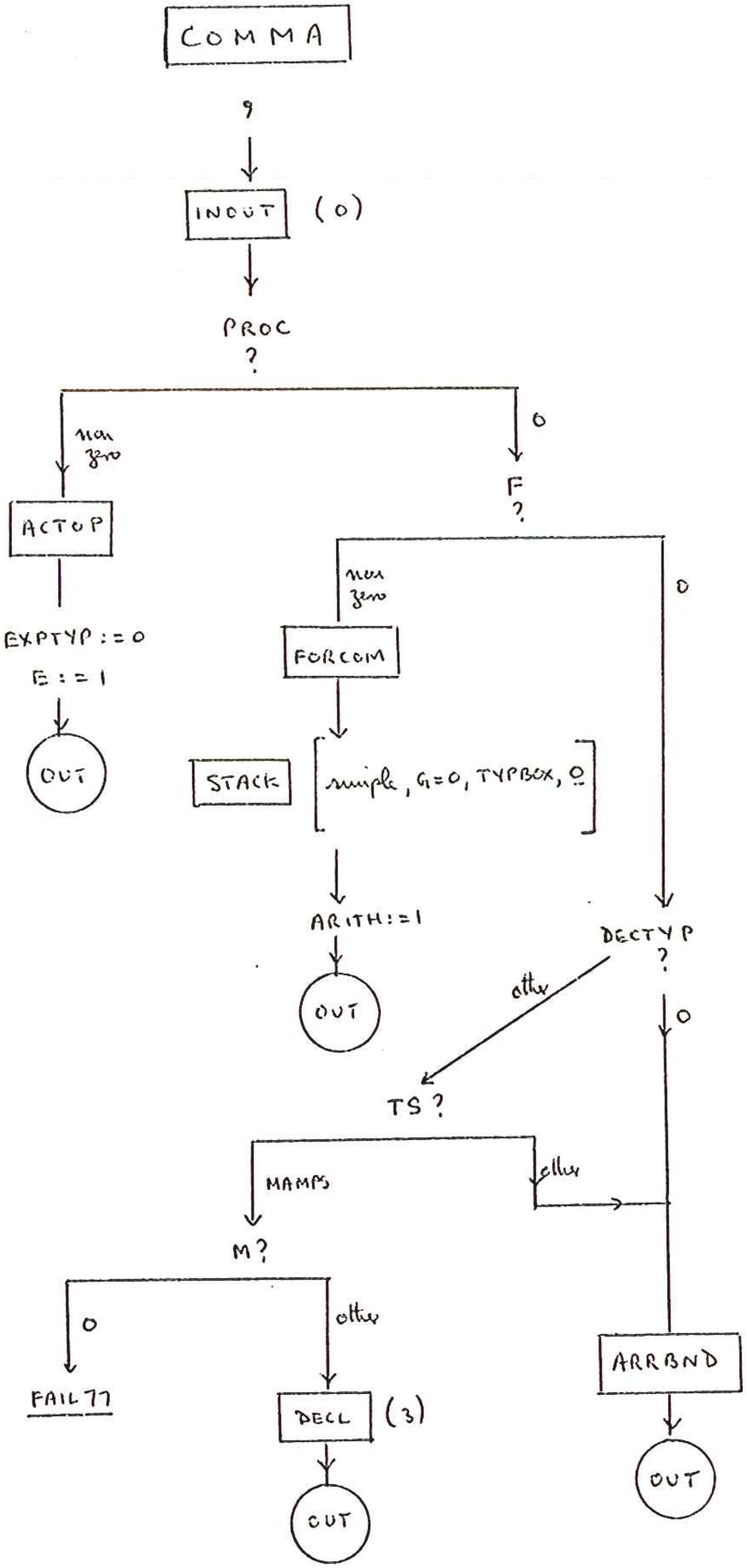
compile shared map pointer (filled in at run time)

ARENT2 is an entry to array

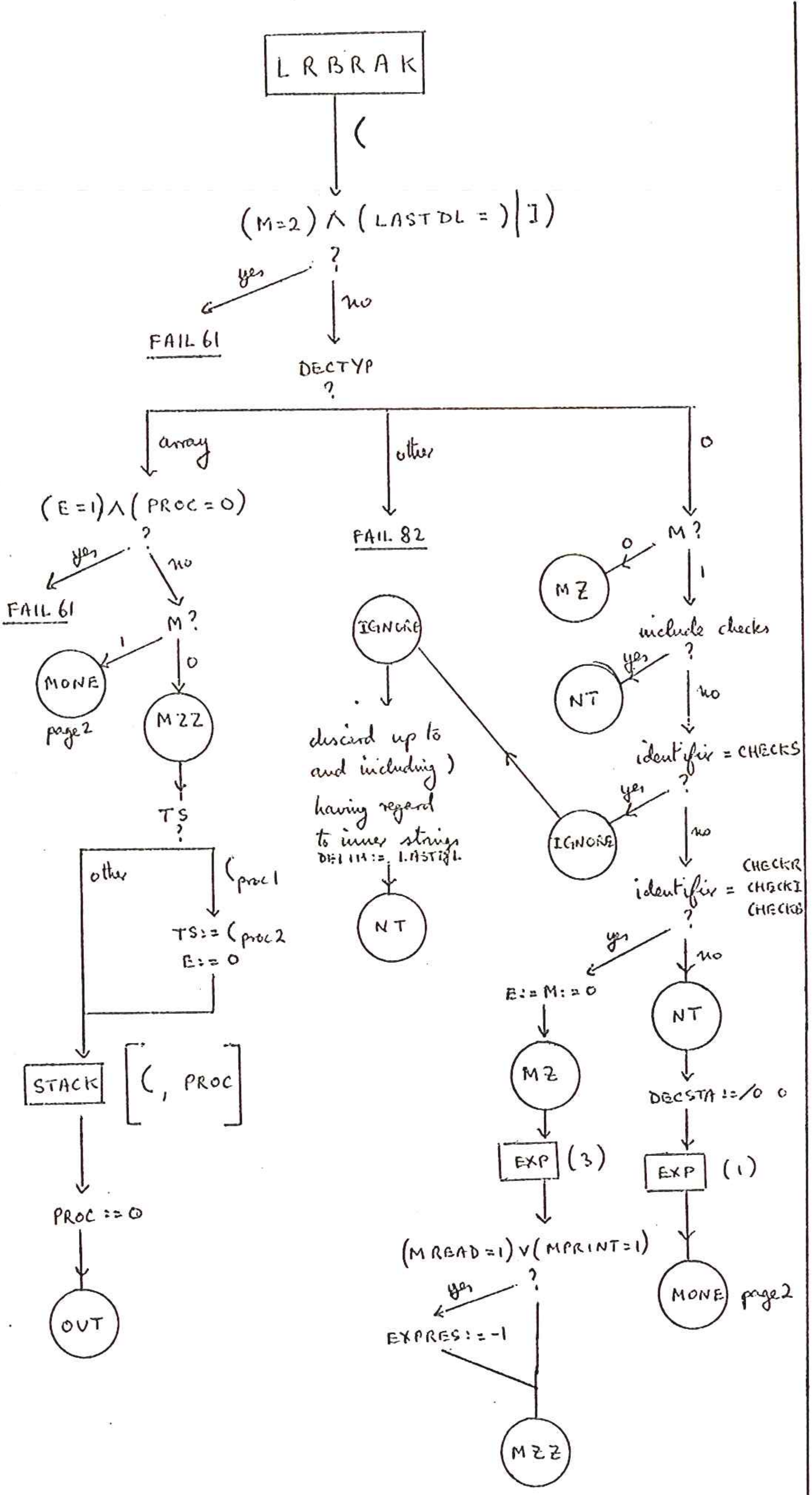


force block name of current block. Check at run time.

Check for overlap with CODL


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

page 1 of 2

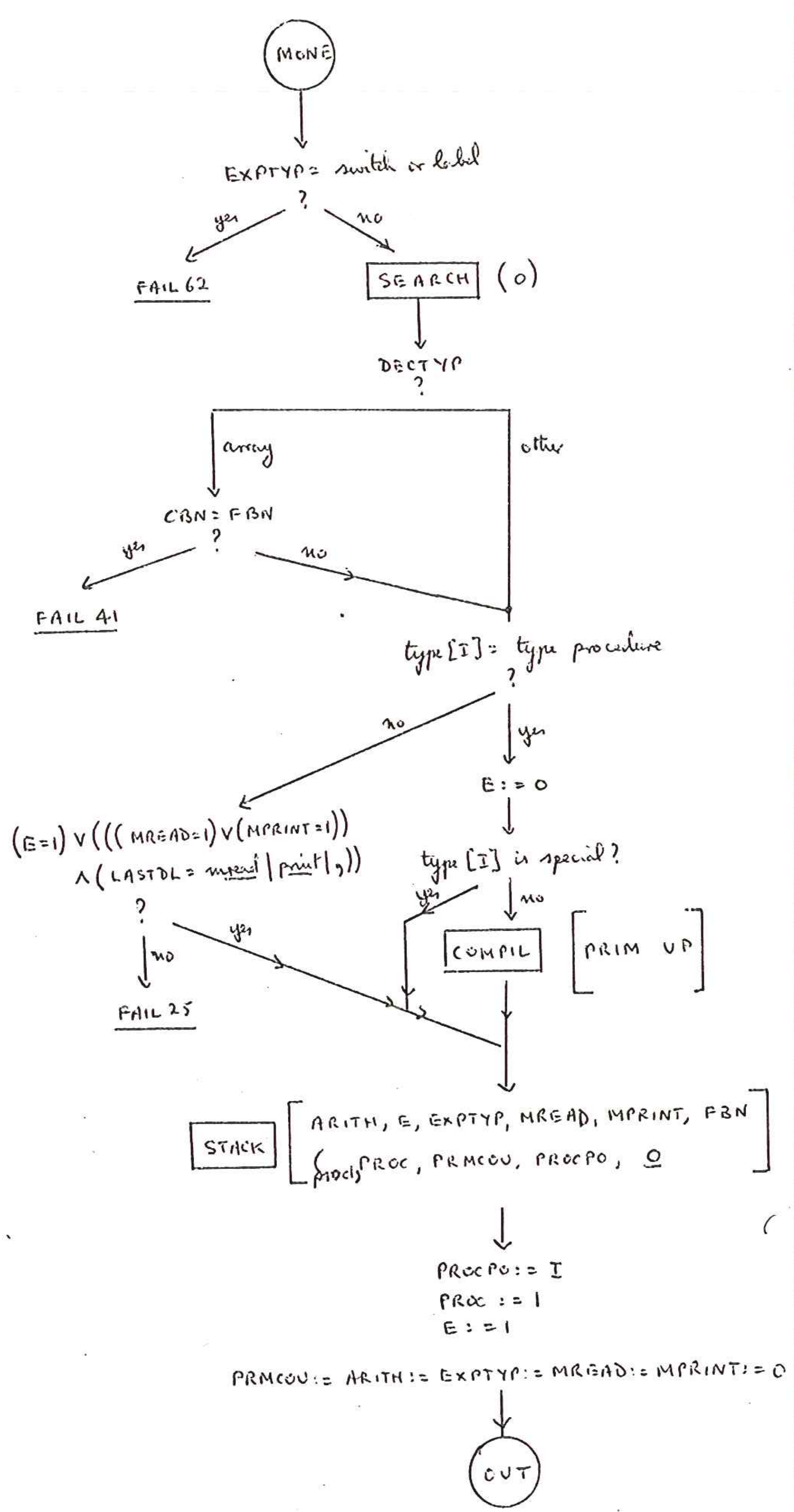


checks included if OPTION 4 bit present

(proc 1 = /2 4096

(proc 2 = /2 5120

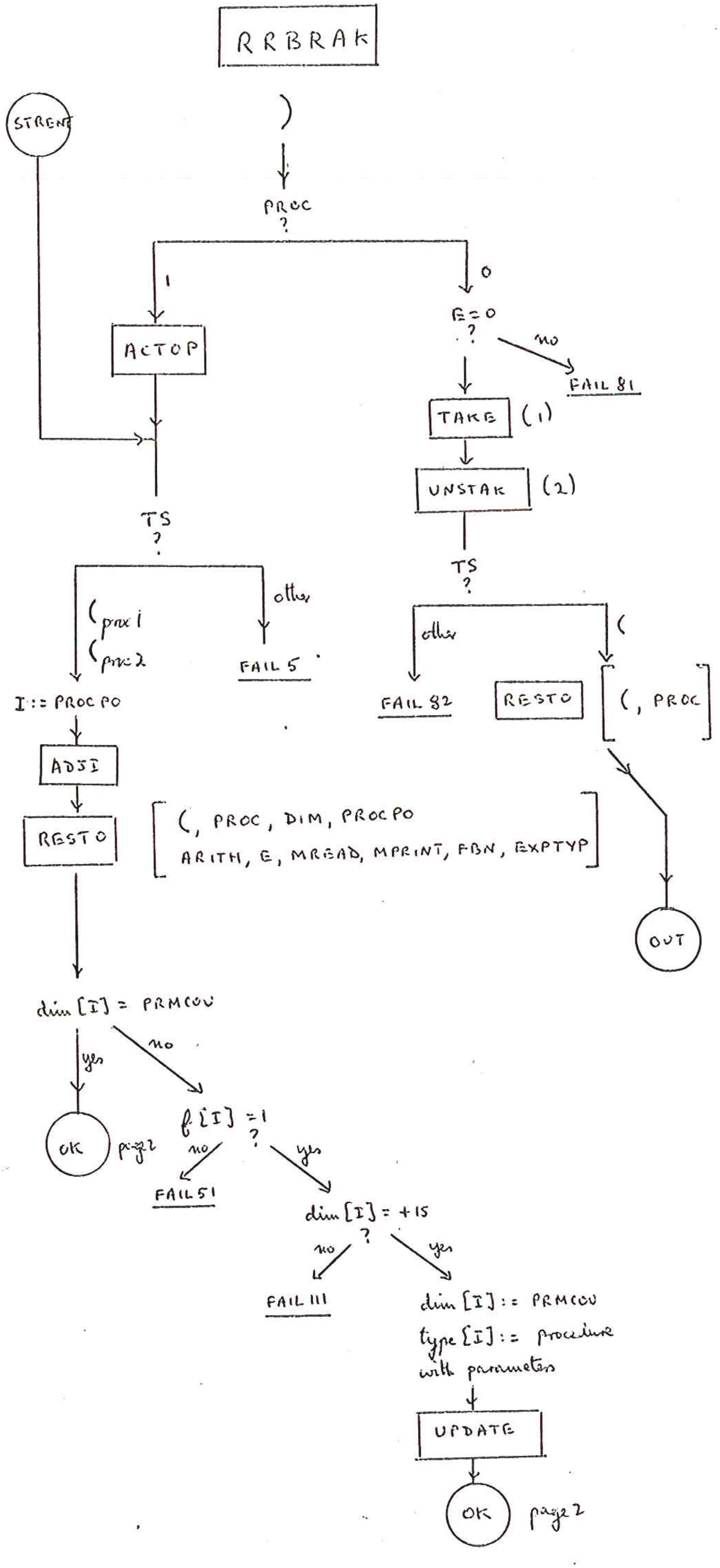
LRBRAK
contd



special means compilable in a PRIM and suppress PRIM UP

(proc1 is/2 4096

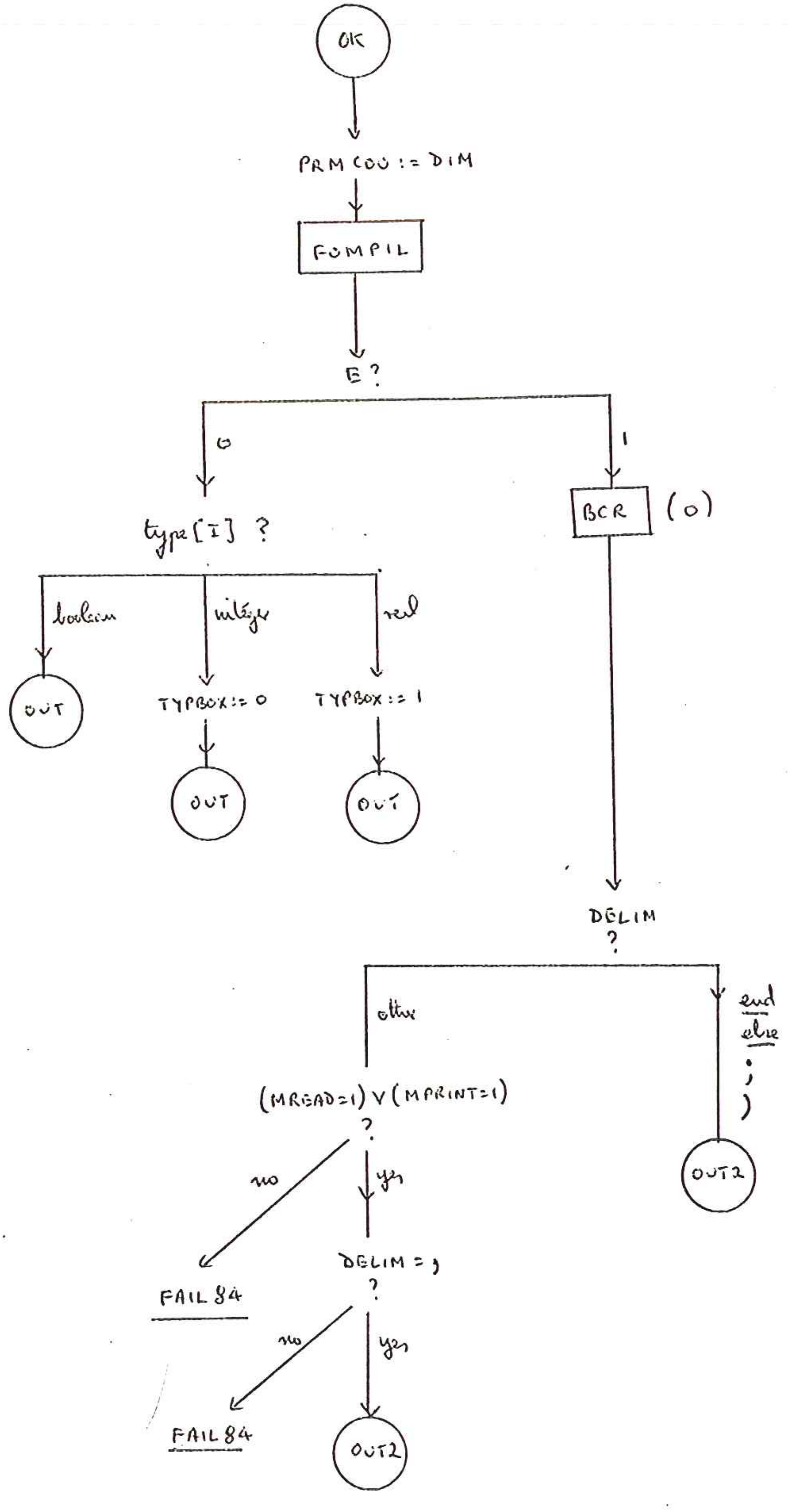
page 1 of 2

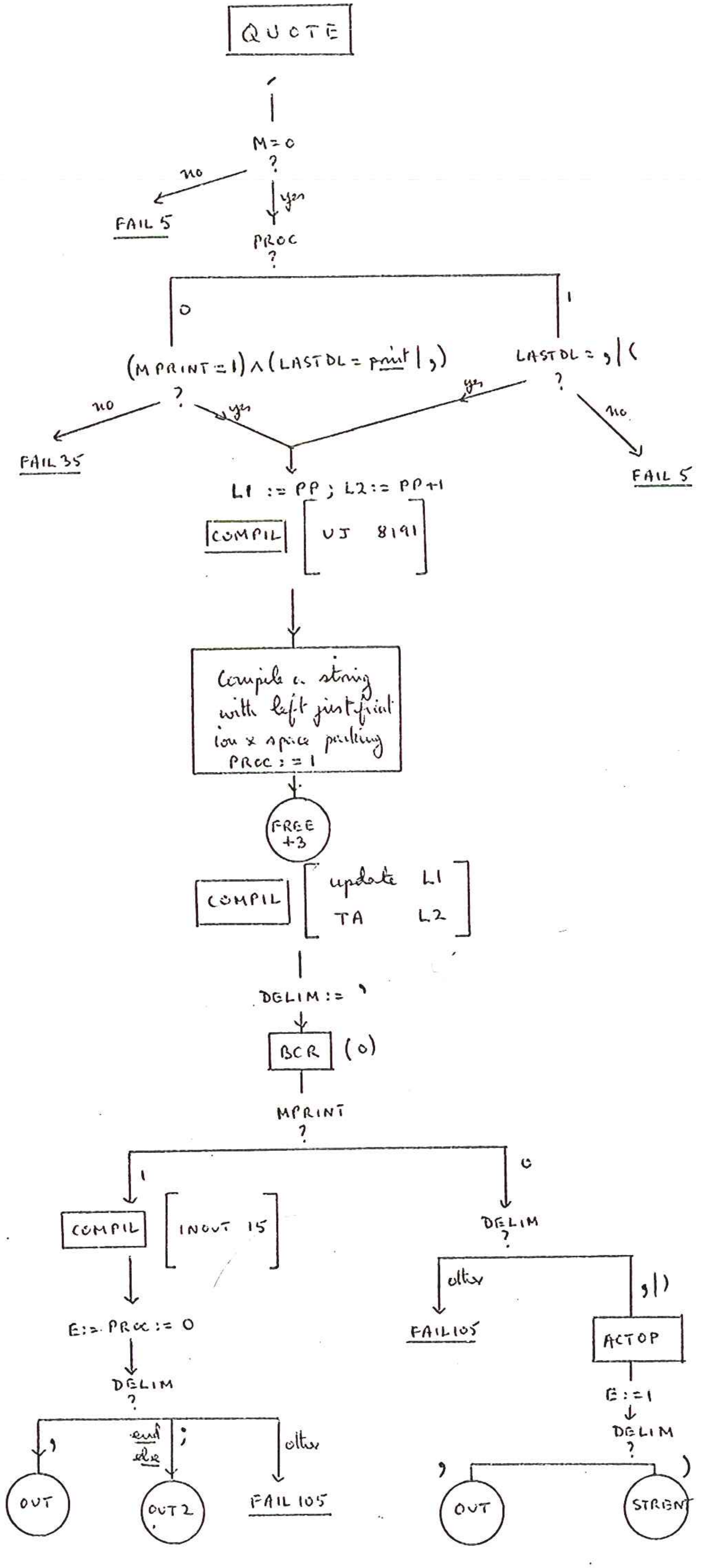


STRENT is an entry from opening string quote

This (may be
(proc1 or
(proc2

RRBRAK
contd.

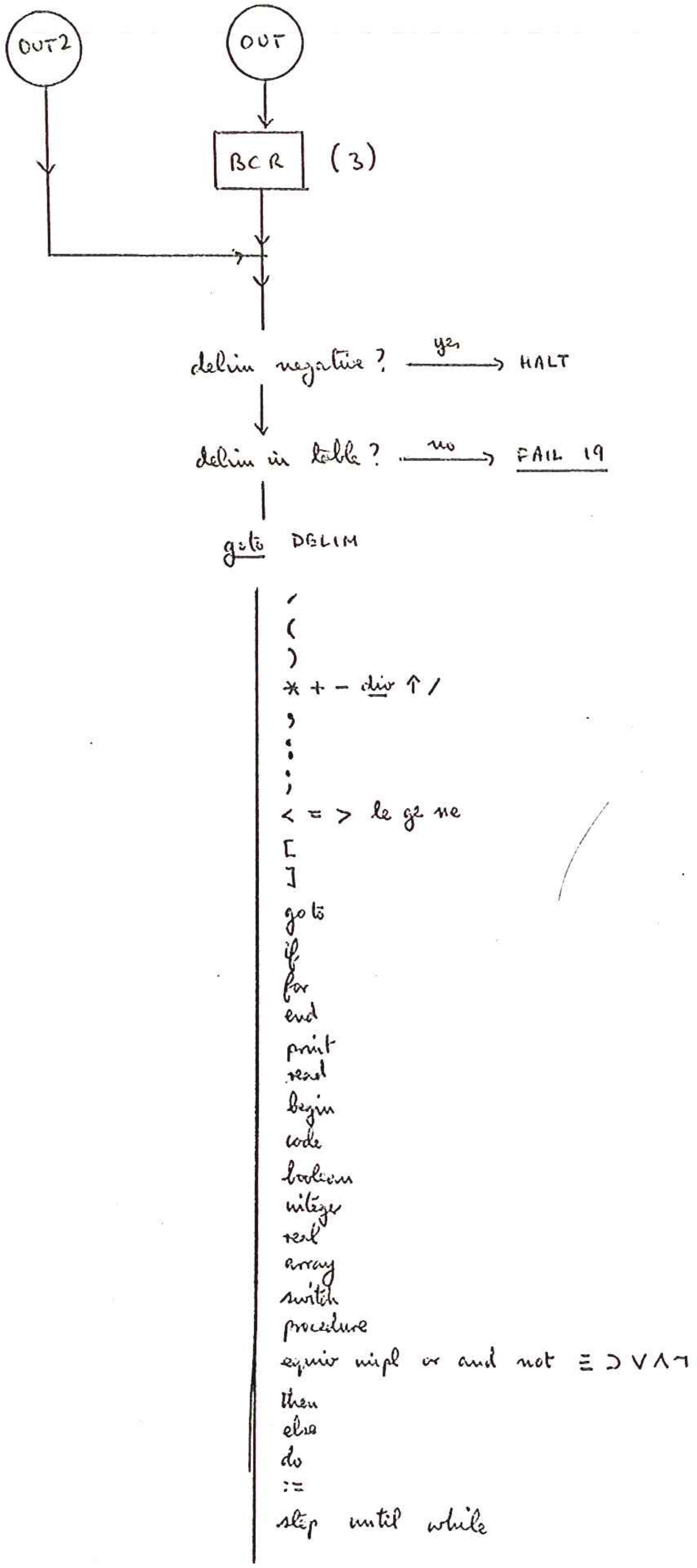




STRENT is entry in RRBRAK)

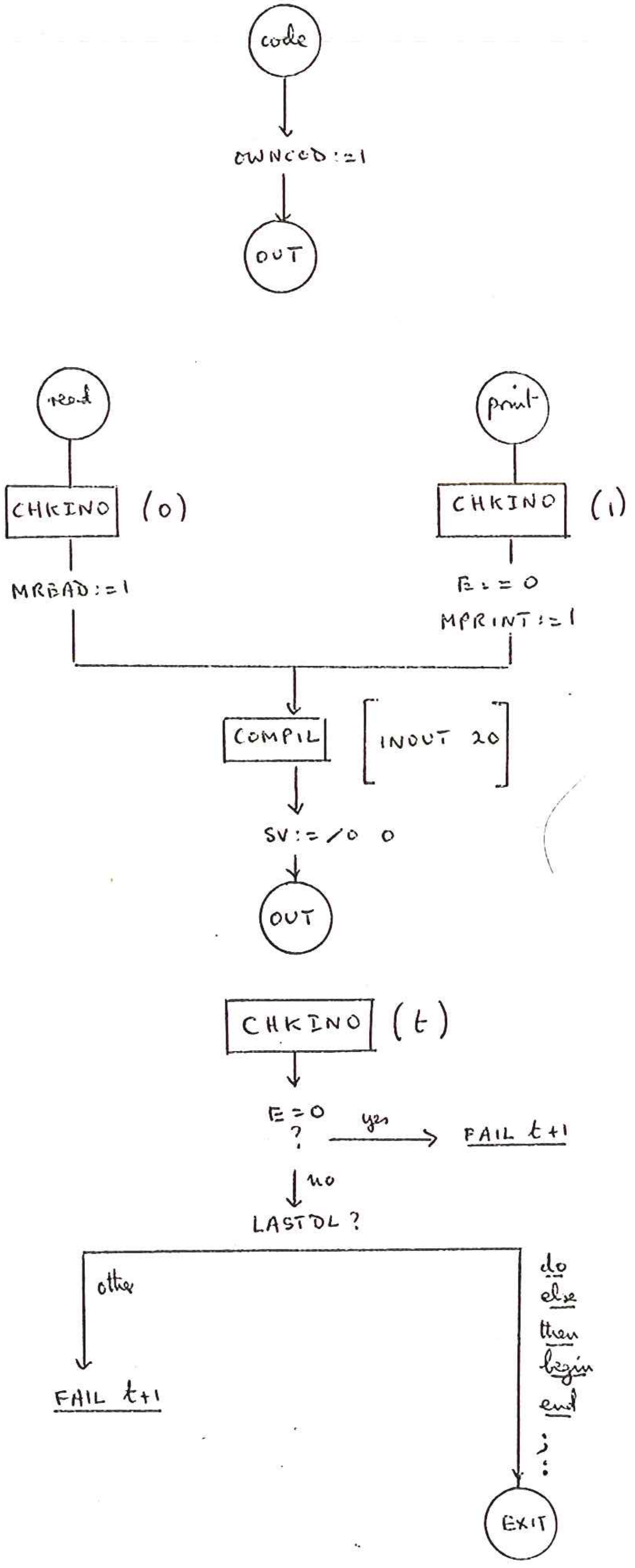
OUT

CENTRAL LOOP



This is impossible to happen. Relic of early testing

code , read , print



precedes owncode declaration and is local to OUT

both local to OUT