THE 903 ALGOL INTERPRETER

CONTENTS

/Contd.

/Contd.

THE 903 ALGOL INTERPRETER

Part 1: Introduction and Basic Routines.

1.1. Introduction.

The Algol Interpreter is used to run a 903 Algol program in the form of object code produced by the Translator. This description should be read in conjunction with the Elliott 903 Algol Object Code Manual (June 1966).

The Translator reads Algol source text, checks it and converts it into an object code program which consists of parameter words (pords), data and workspace. In the basic Algol system for 8K store, this object code is punched in relocatable binary form on paper tape.

The Interpreter reads the program output by the Translator, assembles it into store, and obeys the object code by interpreting the pords in a manner similar to the computer logic obeying machine code instructions.

The Interpreter consists of:-

1) The Algol Loader (based on the SIR relocatable binary loader).
2) The Pord Evaluation routine, which sets up the initial states and then 'obeys' each pord.
3) A set of subroutines.

   To interpret an individual pord, the Evaluation routine selects one of a large number of sub-rountines according to the code value in the pord. These routines (not necessarily written in standard subroutine form) together form the great bulk of the Interpreter. They are subdivided into 3 classes:-

   a) The Arithmetic subroutines, which act on integer, real and boolean values; performing addition, multiplication, exponentiation, equivalence, etc.

   b) The organisation routines, which perform various tasks such as transferring values to and from the stack, setting up stack entries for procedure calls, making branches in the pord

/Contd.

b) Contd.

program, etc.

c) The input-output routines.

1.2. Initial Addresses.

The store from location 8 to location 123 contains a set of entry points, starting addresses and working locations, whose absolute addresses are required to be known. The position of each item is fixed and must not be moved, as any such change would make it necessary to change the majority of all library and users machine code procedures. Certain locations in this range have been left spare, any further items requiring fixed locations may be allocated to these positions or locations immediately above 123. Any new program or additional instruction in existing routines, must be placed at a higher position in store.

A brief description of the store from locations 0 to 123 is given below. For further details the program coding and the individual program descriptions should be consulted.

Address	
0 to 7	These locations are not referred to explicitly at any point in the Interpreter. They are used only implicitly as the S.C.R. and B registers of whatever level the interpreter is obeyed from, normally level 1.
8	Entry to READAL (see 1.3.1.)
9	Entry to CONTIN (see 4.3.)
10	Entry to EXECUT (see 1.4.)
11	Entry to loader (Start C) to input a relocatable binary tape related to the previous tape input. (see 1.3.2.)
12	Entry to LIBENT (see 1.3.3.)
13	Entry to READOL (see 1.3.4.)

/Contd.

Address	
14 & 15	(Spare in Issue 1)
16	Entry to CHNGOP, which sets +4 in STODEV and then comes to a dynamic stop. This causes the lineprinter to be treated as the presumed output device for future entries to EXECUT.
17	Entry to PUNCH 1, which sets +1 in STODEV and comes to a dynamic stop. This restores the punch as the presumed output device.
18 to 20	(Spare in Issue 1)
21 to 25	are reserved for special entry points or addresses for individual users machine code programs.
26	Labelled STKMOD, this location holds either zero or the address at which the stack is required to start. If zero the stack will start at the first free location given by the loader after all programs and data have been loaded. It holds zero in the first version of Algol. If an extra store module was to be used to hold the stack it might be set to +8192 for example.
27	Labelled STKEND, this location holds the address of the block following the area reserved for the stack. Its current value is +8177.
28	Labelled WARNAD, this location holds an address somewhat lower than the beginning of the Algol Loader. If on entry to a procedure or creation of an array the stack pointer exceeds this address WARN is set non zero. (See 1.3.1. for the effect of this). Note that the stack can grow a short distance beyond WARNAD without setting WARN due to ordinary statements. If the stack is not in the first module WARNAD should be set equal to STKEND.
29	Labelled PDADD, this location is used by the loader as the first address for storing Algol pords. It normally holds the first free location after the Algol Library functions.
30	STACKA, is filled by the loader with the next free location after

/Contd.

Address	
30 (Contd.)	any program or data read in.
31	Is labelled BASE, for access to 32 and 33, and also holds the address of the first free location after the interpreter.
32 & 33	are filled by the loader with the addresses corresponding to QACODL and QAVNDA, the Object Data Load (constants) and the ~~Notimal~~ Data Area (variables), respectively. /Notional
34	WARN (See 28, WARNAD).
35	PP (Pord Pointer) holds the address of the next pord to be obeyed.
36	SP (Stack Pointer) holds the address of the 'top' of the stack, the location pointed to and all higher locations are free.
37	EP (Entry Pointer) holds the address of the stack entry made on entry to the current block.
38	FP (Formal Pointer) holds the address of the result space (followed by parameters, if any) of the current procedure. In issue 2 it will only be set for machine code procedures.
39	BN holds the current block and is always an exact multiple of 16.
40 to 55	hold addresses of various routines that are required by library programs. See the program sheets for details.
56 to 65	are special workspace locations required by library programs to be in fixed locations.
66 to 79	are further program addresses and workspace locations required by library programs, plus some spare locations.
80 to 123	are workspace locations which are used by library programs and which may also be used by users machine code subroutines. They may be used freely by machine code procedures so long as these

/Contd.

Address

80 to 123 (Contd.)

do not call interpreter subroutines. If they do call such subroutines the user should consult the subroutine specifications and the program sheets where necessary.

1.3. Special entry points to the Algol Loader.

1.3.1. READAL.

This routine is the most commonly used entry to the loader. It causes the store pointer to be reset to the first free location after the library, thus preserving the library in store but overwriting previous Algol object code programs. The dictionary pointers are set to preserve the Library dictionary only.

Before entering the Loader WARN is tested. If non-zero a previous Algol program has overwritten the Loader. In this case the message 'RELOAD TAPE 2' is displayed and the program exits to STOP.

1.3.2. Entry at 11 (Input related tape).

This entry goes direct to the loader without testing WARN and does not reset the store or dictionary pointers. It must only be used after an entry at 8 or 13.

1.3.3. LIBENT.

This routine tests WARN as in 1.3.1. above. If WARN is zero the Loader is entered at a point which causes the entire dictionary to be deleted and the store pointer reset to the first free location following the Interpreter. A marker is set (in location 75) to indicate that the current tape supplies the Library. This effectively causes the store pointer and dictionary beginning pointer to be preserved and used for subsequent entries to READAL.

/Contd.

1.3.4. READOL.

This entry is used to input a program which does not need the Library or has the necessary Library functions on its own tape. It performs the same operations as LIBENT (1.3.3) except that the marker to indicate Library input is <u>not</u> set. Thus the Library and Library dictionary are overwritten, but no addresses are stored for future use.

1.4. The Pord Evaluation Routine.

This routine commences at EXECUT which sets up various initial states. If BASE +2 is still +8191 a correct Algol program cannot have been loaded, and the routine exits to STOP.

Otherwise the routine sets up PP, SP etc., and the standard state for input-output. The standard presumed settings are placed in the global settings position, taking as output device number the value in STOPEV. This value is +1 for Punch 1 unless set by entry 16 to CHNGOP.

The routine then 'obeys' the first pord by jumping to NXPORD.

NXPORD is entered from EXECUT and from the end of every pord routine in the interpreter. It picks up the word pointed to by PP, stores the least significant 13 bits in ADPART. It increases PP by one, and decodes the most significant 5 bits of the current pord by jumping to a 32 word lookup table FBAJ. This consists of jumps to the basic interpreter function routines, which decode the value in ADPART according to the individual function characteristics.

In the case of the two functions, PRIM and INOUT the address part specifies one of a further set of sub-routines. INOUT is described under the heading of Input-Output. PRIM is specified further below. All other function descriptions may be found by the reference in Section 3.

PRIM.

The address part of pords with function PRIM specifies one of a group of 70 primitive sub-routines, whose addresses are listed in a table PBA ~~P.B.A.~~ Primitives 1 to 29 are entered by a direct jump to the address

/Contd.

listed. Primitives 30 to 70 are written in standard sub-routine form with the link location at the address listed. Before entry to Primitives 30 to 56 W is set equal to SP-6 and SP is reduced by 3. Before entry to primitives 57 and above W is set equal to SP-3.

Primitives 1 to 29 are described in the 903 Algol Object Code Manual, under their individual names. Primitives 30 to 62 are described in Section 2 (the Arithmetic routines).

Primitives 63 to 70 are used to place a check number next to a parameter address. They use a common routine which places (N-60) in location SP-2, where N is the primitive number.

/Contd.

Part 2   903 Algol Arithmetic Narrative

The Algol arithmetic can be conveniently divided into a number of sub-routines, which are entered directly from the pord-evaluator. These sub-routines are labelled PRIM for primitive and each one has a different number associated with it so that each PRIM sub-routine is unique, e.g. PRIM59. Each sub-routine performs one arithmetic operation on either real numbers or integers, or in the case of the functions e.g. LN. for logarithm, on one number only.

Before entering any of these PRIM sub-routines, the pord evaluator uses a location SP, which holds the current value of the stack pointer, to set a workspace location W which then indicates the position in the stack of the number or numbers to be operated on.

The PRIM sub-routines themselves use a few service routines for organisational purposes and some double length arithmetic.

Section 1 of this narrative describes these service routines briefly and section 2 gives a fairly full account of the PRIM S/R's.

2.1. Service Routines

Section 1

The name of the service routine precedes its description.

SINGLE: sets a real number from the stack into w/s locations W3, W4, W5.

FAIL: This is an error S/R which outputs the error number in the accumulator at the time of entry and if continuation is effected sets ± floating point ∞ into ~~and sets~~ W3, W4, W5 dependent on the sign of W3.

RSTACK: Sets 2 real numbers from the stack into w/s locations W3, W4, W5; W6, W7, W8;

RRES: Sets the real number in W3, W4, W5 back into the stack.

SET: Sets boolean results into the stack. Entry at SET+1 sets +0 into the stack indicating _false_. Entry at SET+6 sets +1 into the stack indicating _true_.

STAND: Standardises the real number in W3, W4, W5. If the number is too small for representation or zero W3, W4, W5 are all set zero. If the number is too large then an error is indicated using FAIL (as above).

SHIFT1: Halves the double-length mantissae in W3, W4; W5, W6 and increments their exponents in W5, and W8 respectively.

surely w6 w7 ?

DCMULT: Multiplies together the double-length numbers in W3, W4; W6, W7 and places the double length answer in W3, W4.

ITOR: Changes the integers in W3, W6 to real numbers and stores the standardised results in W3, W4, W5; W6, W7, W8.

CHEBY: Calculates the value of a chebyshev polynomial whose d/ℓ argument is stored in T, T+1.
The number of constants to be used is set in C, and the starting location of these d/ℓ constants, which must be sequential and in order, is set in cstart.
Forms a sequence:
i.e. $B_r = 2 \times arg \times B_{r+1} - B_{r+2} + C_r$

DLDIV: Divides the d/ℓ number in W3, W4 by that in WS17, WS18, and sets the d/ℓ answer in W3, W4.

2.2. Arithmetic Primitives

SECTION 2

The name of the PRIM s/r precedes its description. All real numbers must be in standardised form on entry and are standardised before exit.

PRIM30: Adds 2 integers which are stored in the stack and replaces the first with their sum. If overflow occurs Error 3 is indicated and a dynamic stop obeyed.

PRIM32: Subtracts the second integer in the stack from the first and replaces the first with this difference. Overflow is treated as in PRIM30 above.

PRIM34: Multiplies two integers, held in the stack, and replaces the first with their product. Overflow is as for PRIM30 above.

PRIM38: Raises the first integer in the stack to the power indicated by the second. The result is placed where the first integer was. If overflow occurs error 3 is indicated (see PRIM30 above). If the 2nd integer is -ve then error 20 is indicated and a dynamic stop obeyed. The following results should be noted

$0^X = 0$ for all X

$X^0 = 1$ $X \neq 0$

PRIM41,43,45,47,49,51:

Examines the two integers in the stack and, depending on their values and the particular primitive, replaces the first primitive with a Boolean result. i.e. +1 for True, and +0 for False. There follows a list of the primitive nos. and their functions.

Prim41	If $I_1 < I_2$ then True false otherwise
" 43	$\leq$
" 45	$=$
" 47	$\neq$
" 49	$>$
" 51	$\geq$

PRIM31: Adds two real numbers held in the stack and replaces the first with their sum. If the answer is too large for representation then error 9 is indicated and if continuation is affected then the answer is assumed to be ± Fl. Pt. ∞

PRIM33: Subtracts the second real number in the stack from the first and replaces the first with this difference. Overflow is as for Prim31 above.

PRIM35: Multiples together two real numbers which are held in the stack and replaces the first with their product. Overflow is as for PRIM31 above.

PRIM37: Divides the first real number in the stack by the second and replaces the first with the answer. Overflow is as for PRIM31 above. The following results should be noted

X/0 gives overflow provided X $\neq$ 0

0/X = 0 for all X

PRIM42,44,46,48,50,52:

Examines the 2 real numbers in the stack, and, dependent on their values replaces the first with a boolean result +1 for true, +0 for false. There follows a list of the prim nos. and their functions.

PRIM42 if $R_1 < R_2$ then true, false otherwise.

44	$\leq$
46	$=$
48	$\neq$
50	$>$
52	$\geq$

PRIM53,54,55,56,57:

Examines the 2 boolean numbers in the stack and depending on their values and the prim no. replaces the first with a boolean result, +1 for true +0 for false. There follows a list of the prim nos. and their functions.

PRIM53	If B1 AND B2 then true, false otherwise.
54	OR
55	EQUIV
56	IMPLIES
57	NOT B1

PRIM36: Divides the first integer in the stack by the second and replaces the first with the real answer. The S/R first converts the integers to real numbers and then uses PRIM37 (see back) so that errors and special cases are as for PRIM37.

PRIM39: Raises the first integer in the stack to the power indicated by the second and replaces the first with <u>real</u> answer. Overflow is as for PRIM31 (see back).

The following results should be noted

$0^X = 0$ for all X

$X^0 = +1.0 \quad X \neq 0$

PRIM40: Raises the first real number in the stack to the power indicated by the second and replaces the first with this result. The subroutine uses PRIM61 (log) and EXP (exponential) so that errors which occur in log and exp will also occur here. The following results should be noted:

$0^X = 0$ for all X

$X^0 = +1.0 \quad X \neq 0$

PRIM58: Replaces the real number in the stack by its absolute value.

PRIM59: Replaces the real number in the stack by its entier (an integer). If the result is too large for integer representation then error no. 3 is indicated as in PRIM30 (see back).

PRIM62: Replaces the real number in the stack by
+1 if it is +ve.
+0 if it is zero.
-1 if it is -ve.

PRIM61: Replaces the real number in the stack by its Logarithm. If the number is -ve or zero then error no. 13 is indicated and on continuation the result is assumed to be zero.

EXP: Replaces the real number in the stack by its exponential value. If the real no. is zero the answer is given as +1 immediately. If the real number is >+40 then error 12 is indicated and on continuation the result is assumed to be ± Fl. Pt. ∞ Results that are either too large or too small for representation are treated as in PRIM31.

DIV: Divides the first integer in the stack by the second and replaces the first with the integer result, rounded towards zero if necessary. If the second integer is zero or the result is too large to store as an integer then error no. 3 is indicated as in PRIM30. The ws location W is not set by the pord evaluator in this routine so that on entry the routine uses SP (stack pointer) to set W before obeying the division.

Part 3: The Organisation Routines.

These routines are adequately described in the 903 Algol Object Code Manual under their individual names. For further details the flow diagrams and program listing should be studied. Sub-routines used by the Organisation routines are described in Part. 4. INOUT and the input-output routines are described in Part.5.

/Contd.

Part 4: Subroutines used by the Organisation routines

4.1. FAILEN.

This subroutine prints information when a run time error is found by any interpreter routine.

Entry: Place link in PAUSRT, jump to FAILEN with an error number in the accumulator as a positive interger.

Exit: Exit is standard with W and W3 holding the same value as on entry.

Process: The routine displays the readings:-

ERROR BN PP RETURN followed on a new line by the given error number, the current block name and pord pointer, and the address in the second word of the stack entry for the current block. ~~Each is printed as an integer prefixed by *.~~ The error display is preceded by output of 16 blanks on the punch.

After output the routine comes to a program wait. Since the link is in PAUSRT re-entry at 9 causes exit from the subroutine (see CONTIN 4.3.). If the error is one from which recovery is not possible the subroutine entry should be followed by a stop.

Workspace: The global and local ~~point~~ print sellings are left unchanged by this routine, W3 is reset to its initial value. W1, W2 and W4 to W10, SBW to SBW5; WS and ADPART are all left undefined.

4.2. ENFAIL.

Entry is by direct jump to ENFAIL with an error number in the accumulator. A standard entry is made to FAILEN followed by a stop. This entry is used for non recoverable errors. Another entry is ERROR which is used for compiler errors, e.g. incompatible data. SPARE is entered from all spare positions in the primitive and pord function tables (representing another compiler error).

/Contd.

4.3. CONTIN.

This is entered from the program entry address 9. Its entry parameter is the line in PAUSRT, which is always set before the interpreter enters a program Wait. A jump is made to one plus the address in PAUSRT.

Before exit PAUSRT as set to cause STOP on re-entry, so that if the computer should stop for reason other than a program Wait (e.g. output device in Manual) re-entry at 9 will have no effact. Note that the program can always be restarted by entry at 10.

4.4. STSUBRT (Assign).

This subroutine assigns a given real or integer value to a given address, with optional packing if the value is real.

Entry: Store the link in STSUBRT.

Enter at:-

1) STSUBRT +1 with the address at (SP) -6 and the value at (SP) -3.

2) STSUBRT +7 with the address at (PKDADD) and the value at the address held in the accumulator.

3) STSUBRT +15 with the address at (PKDADD) and the value in W3, W4, and W5.

Exit: Exit is standard if entry (1) is used.

SP: = SP-3 and W: = SP-6

Process: If the given address is that of a constant (bit 17=1) a non-recoverable error indication is given. If the address is that of a real value (bit 18=1) the given value in 3 locations is packed into 2 locations, with rounding to the nearest 2^{-27}. If the exponent is less than $^{-64}$ zero is assigned, if greater than $^{+63}$ the error for floating point overflow is given and $\neq$ 0.99999999* 2^{63} is assigned. If the location following the address is negative, the number is assigned without packing

4.5. W34SR.

This subroutine copies a real value (in unpacked form) from a given

/Contd.

position into locations W3, W4 and W5.

Entry: Store link in SBLNK and use entry:-

1) W345R copies the topmost value in the stack.
2) W345R1 copies the value whose address is held in the accumulator.
3) W34R2 as for (2) with address in B register.

Exit: Standard.

Workspace: W3, W4, W5. Also W if entry (1) or (2) used.

4.6. RICONV.

The subroutine converts the real number in W3, W4, W5 to integer form. The process used gives the Algol automatic type conversion (integer: = enter (real + 0.5)).

Entry: Store line in SBLNK1 and enter at RICONV.

Error: If the number is greater than 131071.5 or less than -131071.5 the error for integer overflow is given. Continuation is not possible.

Exit: Standard, with the integer in the Accumulator and W3.

Workspace: W3, W4, W5, SBW.

4.7. ITRSB1.

This subroutine converts an integer to a real value. The real value is in standardised floating point form, unpacked in locations W3, W4 and W5.

Entry: Store link in SBLNK, enter at:

1) ITRSB1 with the integer in the Accumulator.
2) ITRSB2 with the integer in W3.

Exit: Standard.

Workspace: W3, W4, W5, SBW.

/Contd.

4.8. NEGR1.

This subroutine negates a given real value in standard unpacked floating point form.

Entry: Store link in SBLNK, enter at:-

1) NEGR1 with the address of the real value in W.
2) NEGR6 with the address in the B register.

Exit: Standard, with the negated number in the original position.

Workspace: Entry (1) W. Entry (2) none.

4.9. FINDFP.

This subroutine finds the formal pointer corresponding to a given block number.

Entry: Standard subroutine entry, place link in FINDFP and enter at FINDFP +1. ADPART contains the given block number BN^{1} and n (the procedure parameter number) in the least significant 4 bits.

Exit: Exit is standard, with $3n + FP^{\ell}$ in the Accumulator, where FP^{ℓ} is the formal pointer corresponding to BN^{ℓ}.

Errors: If the given block name is not found a "compiler error " is printed (recovery not possible).

Workspace: SUBWK1, SBW1.

4.10. GARAD.

This subroutine gets the address of an array map, given the address of the array.

Entry: Store link in SBLNK and enter at GARAD with the address in the Accumulator.

Exit: Standard, with the array address in W and the map address in W1.

Workspace: W, W1.

/Contd.

4.11. AINDSB.

This subroutine finds the absolute address of an array element, given the indices in the stack. It is used by INDA and INDR pord functions.

Entry: Standard, place link in AINDSB and enter at AINDSB +1. ADPART holds n, the number of dimensions, and the indices are held in stack at the positions SP -3n, SP -3n + 3,....., SP-3. The array address is at SP-3n-3.

Exit: Standard, with the element address in the Accumulator. If a real array, bit 20 has value 1.

Error: If the array element falls outside the limits of the array a non-recoverable error is given. (Array index wrong).

/Contd.

Part 5: The Input-Output Routines.

5.1. INOUT.

This routine is used to call a further set of subroutines, depending on the value in the address part of the pord. It determines whether the Inout number is greater than 15 to inform the setting procedures that a local or global setting is required. The address of the particular Inout routine required is obtained from a table at INOUT 3. All the routines are entered as subroutines with a common link at IOLNK. However, only numbers 1 to 4 make use of IOLNK on exit, all the others exit direct to NXPORD.

5.2. Character input and output.

All input and output via paper tape station is done through the subroutines GECHAR or OTCHA. (This includes output to the teleprinter and line printer where fitted.). The only exceptions to this rule are the output of blanks to the punch in FINISH and FAILEN and the input of tape by the Algol Loader.

5.2.1. OTCHA.

This subroutine outputs one character to the current output device. It is identical in the 920 version to the standard 903 version.

Entry: Store link in OTCHA and enter at OTCHA 1 (OTCHA +1), with the character to be output held in the Accumulator and the required output device number held in ODEV.

The character to be output is either:-

(a) A six-bit 903 Internal code character.

(b) A binary pattern to be output, with the sign bit set to one.

Entry to OTCHA2 causes a repetition of the previously output character.

/Contd.

X

Exit: Exit is standard, with ODEV unaltered.

Workspace: SBW.

Method: If the sign bit is not present, the output code value of the character is found from TABLE. If a double character is indicated, the character in VBARCH is output before the translated character. (This is redundant in the 903 version). The output is done by using ODEVTB [ODEV] to modify a/15 0 instruction.

If the character is internal code newline, the character from TABLE is followed by output of the character from LFCH and a blank. (In the 920/503 version LFCH is blank also).

A test is included for output device numbers greater than 4. At present this merely causes a jump to exit from the subroutine. At a later date this may be used to enter a digital plotter output routine and/or other special output device routines.

5.2.2. GECHAR.

This subroutine effectively gets one character from the current input device. Different versions are provided for the 903 and (920/503 code) Interpreters. Entry and exit are identical for both versions.

Entry: IDEV must contain the required input device number.

Enter at GECH with the link stored in SBLNK.

or Enter at GECHEN with the link stored in GECHAR.

Exit: Exit is standard. The character just read from tape and translated to internal code is held

/Contd

in BUFFER [IDEV].

The previous character (in internal code) is held in NEXTCH. The group code for this character is held in the Accumulator on Exit. (For a description of groupcode see the Translator description, page 21). (For the 920 Interpreter substitute "character just taken from the input buffer" for "character just read from tape").

<u>Method:</u> (1) 903 Interpreter.

A character is taken from BUFFER (IDEV) and stored in NEXTCH. If this character was marked with the sign bit it represented a halt code and a wait stop is entered. (The initial routine EXECUT ensures that 'space' is the first character taken at the start of a program).

The program then reads one character from the current input device. A/15 0 instruction is modified by IDEVTB [IDEV]. Seven bits of this character are used to look up TABLE values. The parity bit is compared with that from the table. The least significant 6 bits of the table look up give the internal code value, except for special characters. Special characters are grouped into: illegal characters, ignorable characters, newline and halt.

The appropriate internal code is stored in BUFFER [IDEV]. The groupcode TABLE [NEXTCH] is taken before exit from the subroutine.

<u>Method:</u> (2) 920 Interpreter.

A character is taken from BUFFER (IDEV) and stored in NEXTCH. BUFLAG is then tested. If

/Contd.

negative the line input buffer is empty. (EXECUT sets BUFLAG negative at the start of each program).

If the line buffer is not empty the next character is picked up and stored in BUFFER [IDEV]. The groupcode for NEXTCH is found by looking up TABLE and the routine exits. If the character picked up from the line buffer was newline then BUFLAG is set negative and HALTMK tested. If HALTMK is true the newline character represents a halt code input and S^WAIT (Systems wait) is entered.

If the line buffer is empty characters are read from paper tape, converted to internal code, and packed 3 to a word until the next newline is read. Vertical bar is treated as a special character, the following significant character is read and the appropriate internal code formed. Halt code is treated in the same way as newline, with a special setting of HALTMK to true. If an illegal character is found the buffer if filled up to newline before an error is given.

When the line buffer is full the routine returns to pick up the first character from the buffer and exit in the normal way.

5.3. Output number routine.

Function: To print a real number or integer on a specified output device, using a specified 903 Algol format.

Entry: In all cases the link is stored in IOLNK.

1) Enter at OUTR to print the real number at (W) using the current local print settings. A holds th

/Contd.

new value of SP.

2) Enter at OUTI to print the integer at (W) using the current local print settings. A holds the new value of SP.

3) Enter at OUTR2 with the required print mode given in the Accumulator to print the real number in (W). The output device number must be set in ODEV and the required digit settings in DIGM and DIGN where appropriate:

Output mode	Accumulator	DIGM	DIGN
Freepoint (n)	+ 0	(not used)	+n
Aligned (m,n)	- 1	+m	+n
Scaled (n)	+ 1	(not used)	+n

4) Enter at OUTI4 to print the integer held in the accumulator. ODEV must hold the output device number required. The current digits number is taken from INTDG.

5) Enter at OUTI5 to print the integer held in W3. ODEV must be set as for (4) and the accumulator must hold +n for a print in the style: digits (n).

Exit: Exit is to INOEX, which causes a standard subroutine exit via the link in IOLNK.

<u>Method</u>: Integers are converted by the standardise routine and printed as real numbers in the form aligned (n,0). The real number is converted from the form:

$N = a * 2^b * 10^0$ where $-1.0 \leqslant a < -0.5$ or $+0.5 \leqslant a < +1.0$

to:

$N = F * 2^0 * 10^x * SIGN$

where $+1.0 > F \geqslant +0.1$
X is an integer
SIGN is +1 or -1

/Contd.

The modulus of N is first taken, then X is found by repeated multiplication or division by +10.0, until F is in the required range when b is reduced to zero.

If the number cannot be printed in the specified format an alarm print takes place according to the rules in the 903 Algol Manual. Otherwise the number is printed by OTMXD as a mixed number. If in scaled format the number is printed as $F * 10^1$ and followed by subscript 10 and (X-1) printed as an integer with sign and non-significant zeros.

Working locations: W, ODEV, are not altered by the routine. W3, W4, W5, DIGM, DIGN, SBLNK, SBW, SBW1, SBW2, ADPART, SBLNK2, WS2 to WS8, WS14, NSIGNF, SIGNCH, TENPWR, SW, W1, W2, W7, W8, W9, WS10, WS11, WS12, are altered in an undefined manner.

5.4. Input number routine.

Function: To input a number punched in one of the standard Algol forms from the current input device.

Entry: Store link in IOLNK and enter at RDNM. To read real number ADPART must be set +2, otherwise an integer is assumed. W points to the address to which the number is assigned.

Exit: The number is assigned to the given address. SP is set equal to W. The number may be found in W3, W4 and W5 (W3 for integer). Exit is via INOEX which causes a standard return ~~to~~ via the link in IOLNK.

Method: The routine always reads a real number, converting to integer form before exit if required. Each digit input is mul~~tiplied by 10~~ scaled and added to a double length mantissa, which is multiplied by 10.0 before the addition.

/Contd.

A count is kept of the number of digits after the decimal point and this is combined with the decimal exponent (read as an integer). The floating point number formed from the digit input routine is then multiplied or divided by 10.0 the appropriate number of times.

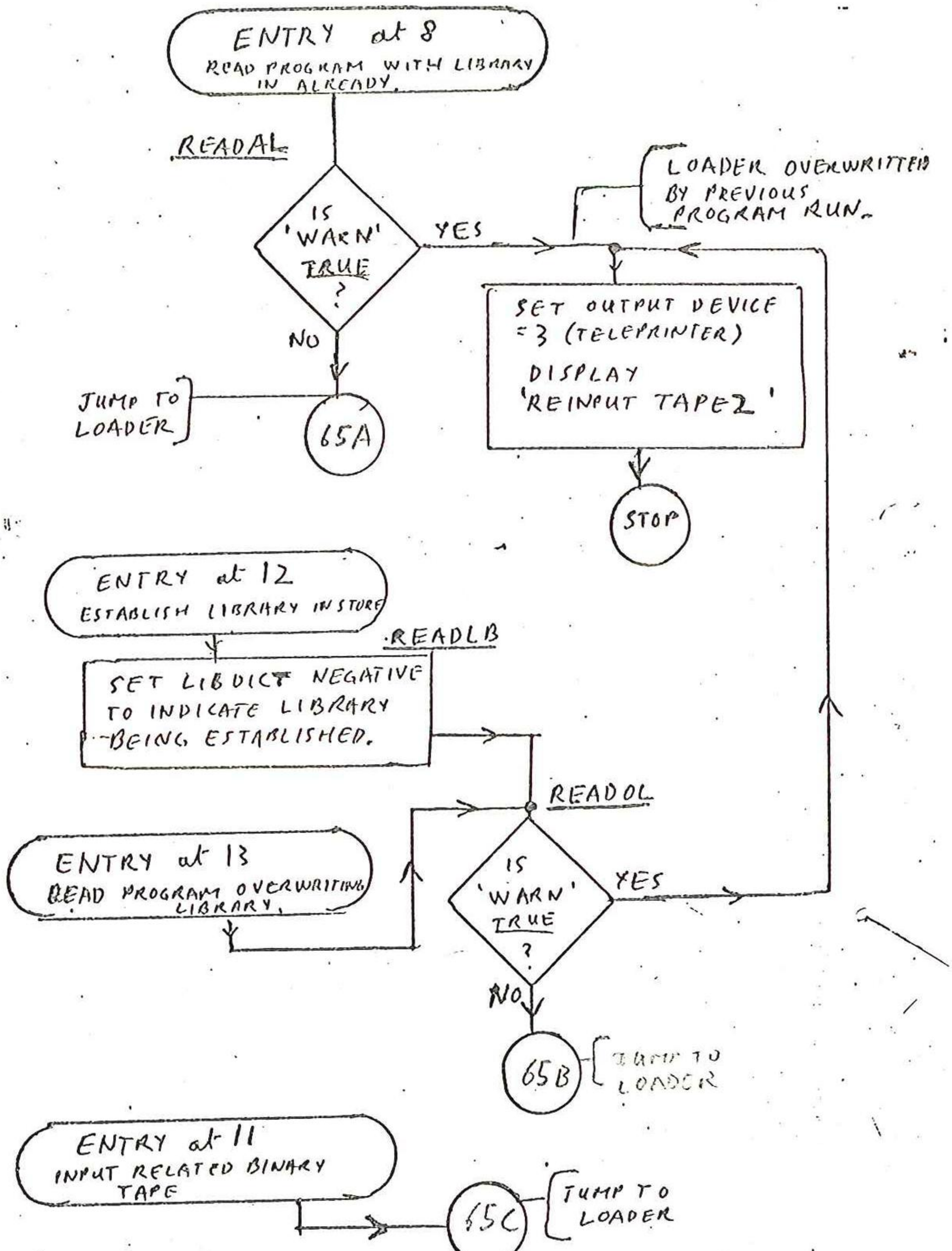
Before exit the number is negated if a negative sign was read, and converted to integer form if required.

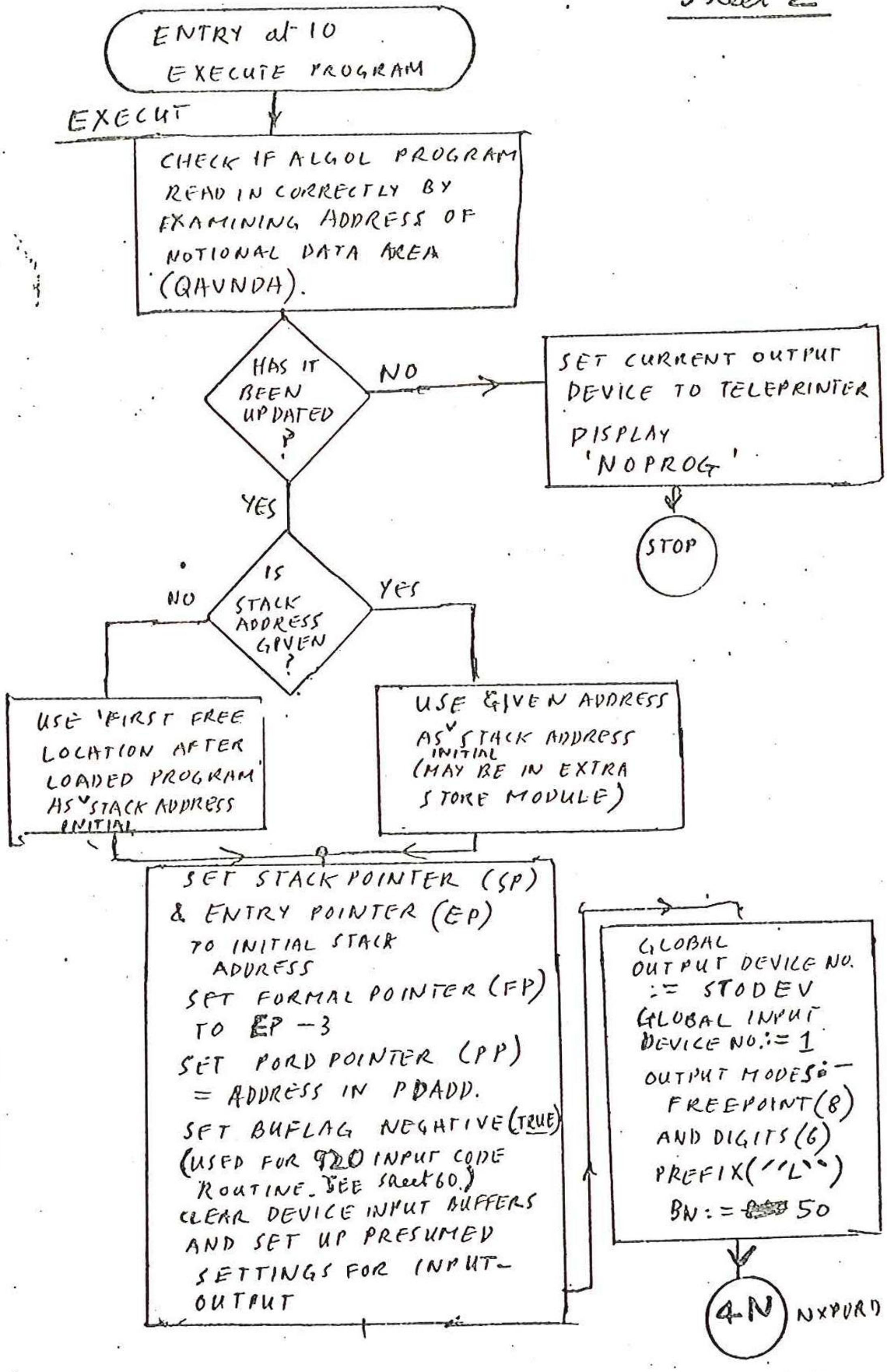
5.5. Other INOUT routines.

These routines take their parameters from the top of the run-time stack. For read and point settings one routine is used for both global and local settings, the difference having been detected by INOUT. These routines detect whether the settings are out of range, and where necessary replace them by the standard settings. Thus the read and point routines do not have to allow for out-of-range settings.

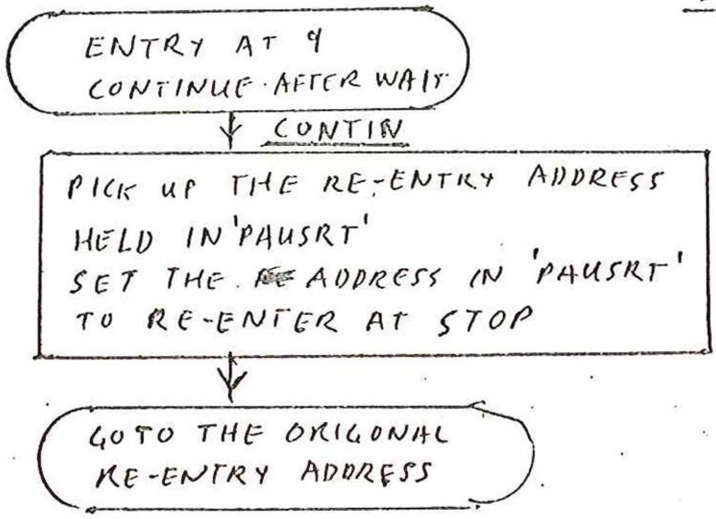
For further details the 903 Algol Object code Manual, the flow diagrams and program sheets should be studied.

903 ALGOL INTERPRETER Sheet 1




ENTRY at 10
EXECUTE PROGRAM
EXECUT
CHECK IF ALGOL PROGRAM READ IN CORRECTLY BY EXAMINING ADDRESS OF NOTIONAL DATA AREA (QAUNDA).
HAS IT BEEN UPDATED ?
NO
SET CURRENT OUTPUT DEVICE TO TELEPRINTER
DISPLAY 'NOPROG'
STOP
YES
IS STACK ADDRESS GIVEN ?
NO
YES
USE 'FIRST FREE LOCATION AFTER LOADED PROGRAM' AS INITIAL STACK ADDRESS
USE GIVEN ADDRESS AS INITIAL STACK ADDRESS (MAY BE IN EXTRA STORE MODULE)
SET STACK POINTER (SP) & ENTRY POINTER (EP) TO INITIAL STACK ADDRESS
SET FORMAL POINTER (FP) TO EP -3
SET PORD POINTER (PP) = ADDRESS IN PDADD.
SET BUFLAG NEGATIVE (TRUE) (USED FOR 920 INPUT CODE ROUTINE. SEE sheet 60.)
CLEAR DEVICE INPUT BUFFERS AND SET UP PRESUMED SETTINGS FOR INPUT-OUTPUT
GLOBAL OUTPUT DEVICE NO. := STODEV
GLOBAL INPUT DEVICE NO. := 1
OUTPUT MODES:-
FREEPOINT (8)
AND DIGITS (6)
PREFIX (''L'')
BN := 50
4N
NXPORD

Sheet 3



Room for further extensions, e.g level change
program and level 3 stimulus

Sheet 4

NXPORD

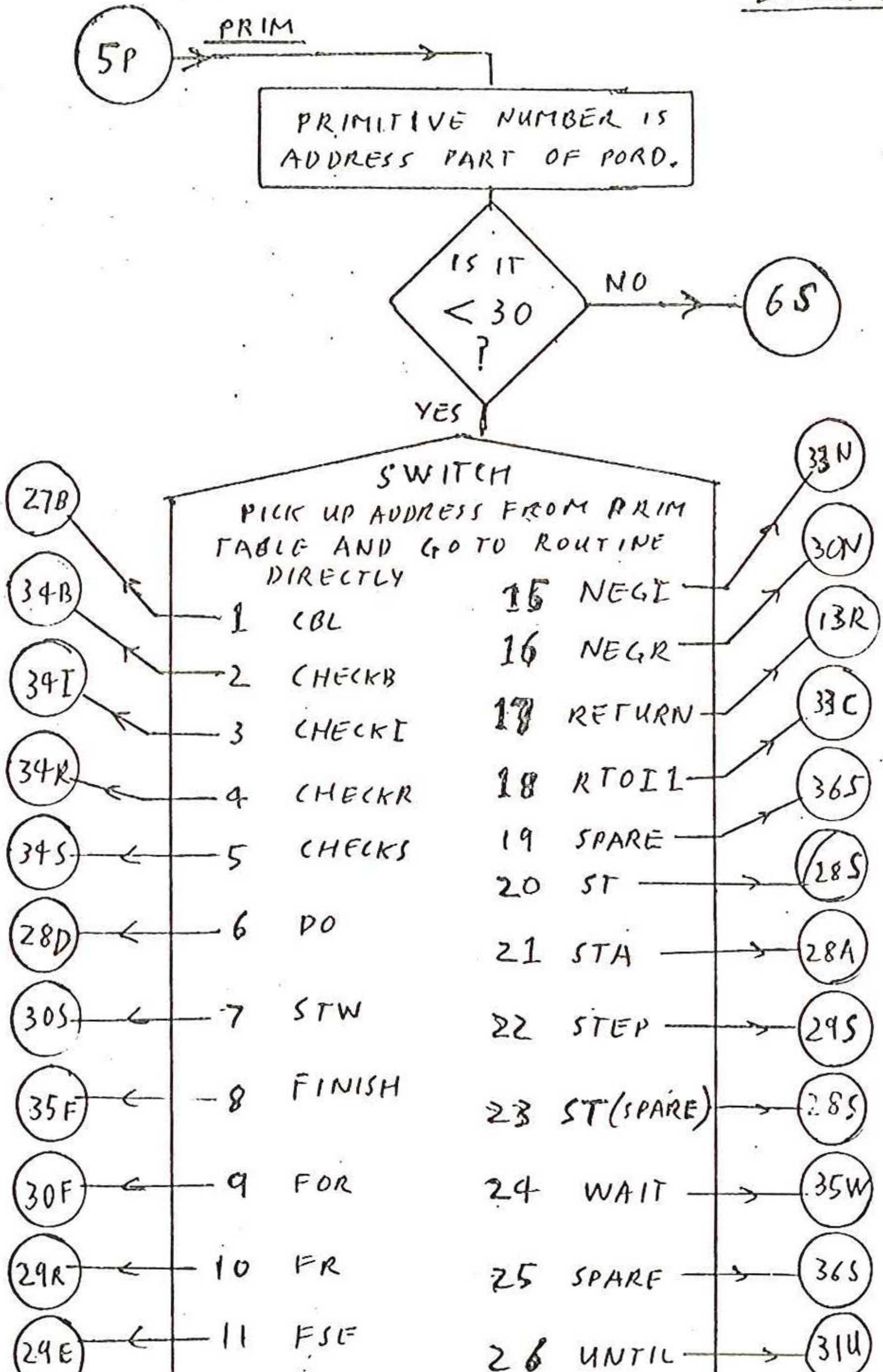
4N

PICK UP WORD INDICATED BY PORD POINTER.
ADVANCE PORD POINTER BY ONE.
PICK OUT ADDRESS AND FUNCTION PARTS FROM PORD. (STORE ADDRESS IN ADPART)

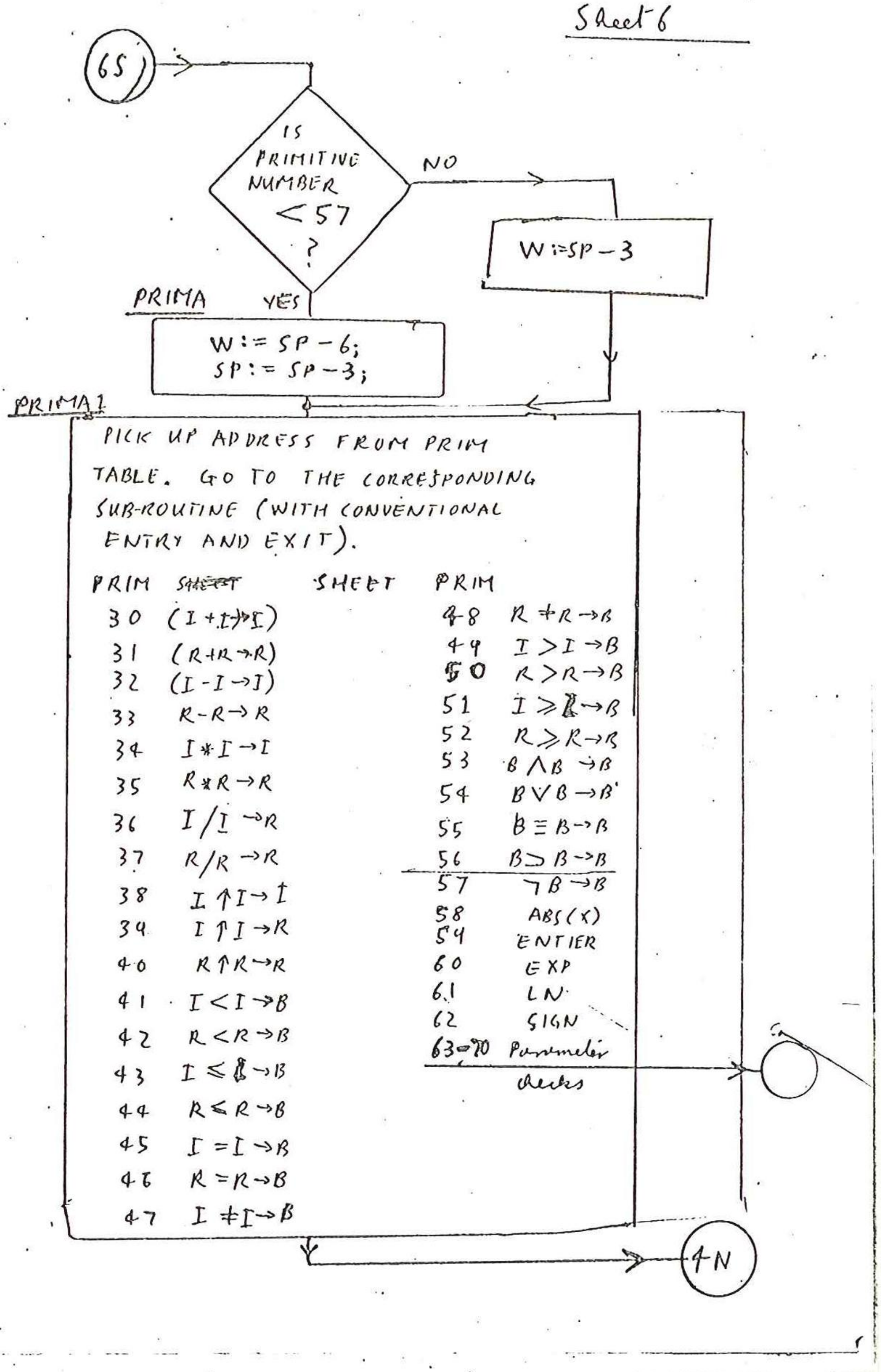
SWITCH
"GO TO" ACCORDING TO FUNCTION PART

14T	0	TA	16	TRAP	14P
14N	1	TIA	17	TICA	14C
15V	2	TIR	18	TIC	16C
14R	3	TRA	19	TRCA	14D
15P	4	TRR	20	TRC	15K
36S	5	INDFS ~~SPARE~~	21	CF	27P
23M	6	MAMPS	22	CFF	27F
16F	7	IFJ	23	PE	17P
16J	8	UT	24	TFAI (TF)	22F
21S	9	GTS	25	TFVR (SPARE)	22F
20G	10	GT	26	TRCN	22N
20F	11	GTF	27	INDS	26S
26A	12	INDA	28	IFUN	15F
26R	13	INDR	29	RFUN	15R
21F	14	GTFS	30	PEM	13P
36I	15	INOUT	31	PRIM	5P

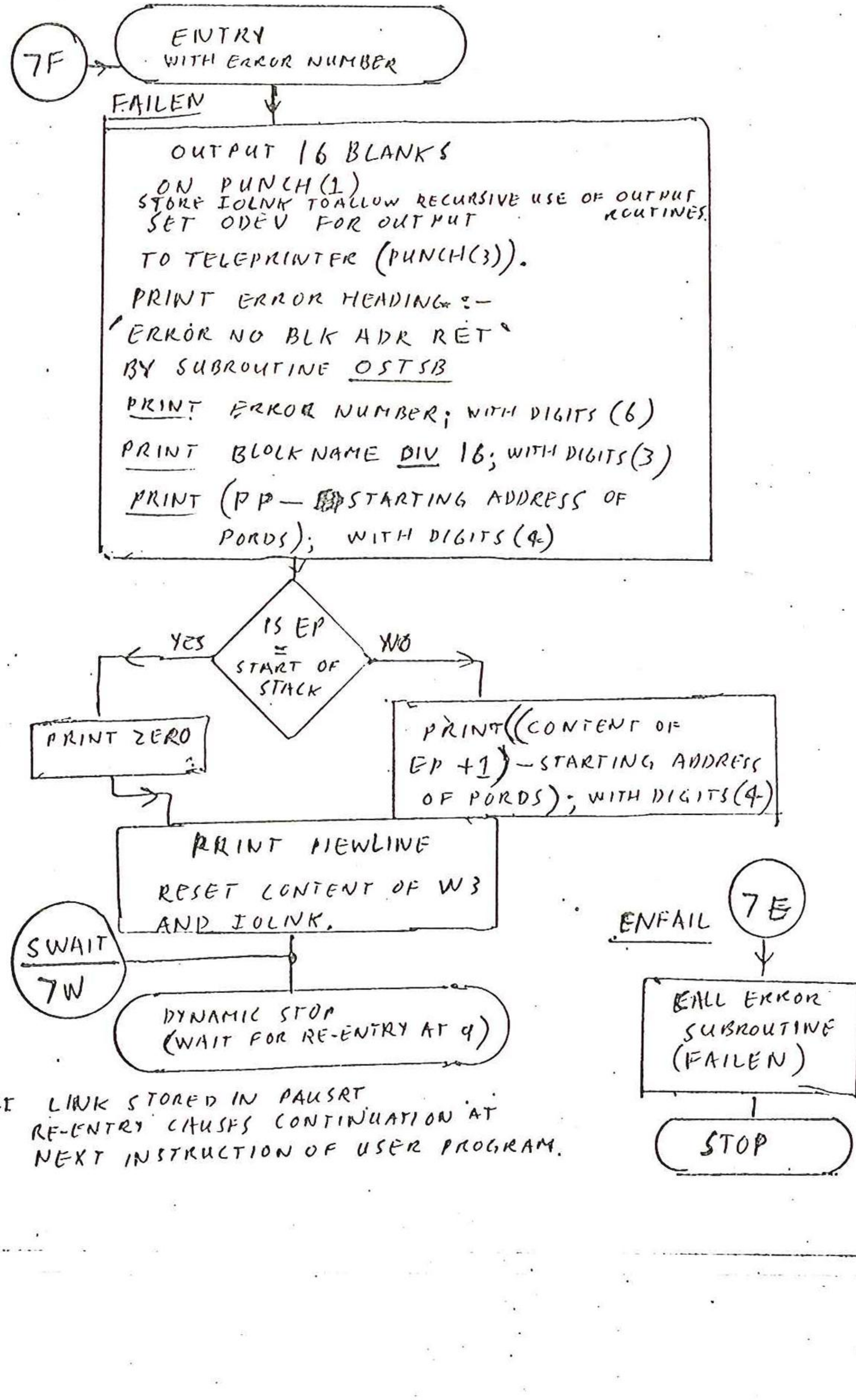
Sheet 5


5P
PRIM
PRIMITIVE NUMBER IS ADDRESS PART OF PORD.
IS IT < 30 ?
NO
6S
YES
SWITCH
PICK UP ADDRESS FROM PRIM TABLE AND GO TO ROUTINE DIRECTLY
27B 1 CBL
34B 2 CHECKB
34I 3 CHECKI
34R 4 CHECKR
34S 5 CHECKS
28D 6 DO
30S 7 STW
35F 8 FINISH
30F 9 FOR
29R 10 FR
29E 11 FSE
12 DIV
33A 13 ITOR1
33B 14 ITOR2
15 NEGI 33N
16 NEGR 30N
17 RETURN 13R
18 RTOI1 33C
19 SPARE 36S
20 ST 28S
21 STA 28A
22 STEP 29S
23 ST(SPARE) 28S
24 WAIT 35W
25 SPARE 36S
26 UNTIL 31U
27 UP 14U
28 PRIM28(RAI)
29 WHILE 32W

Sheet 6

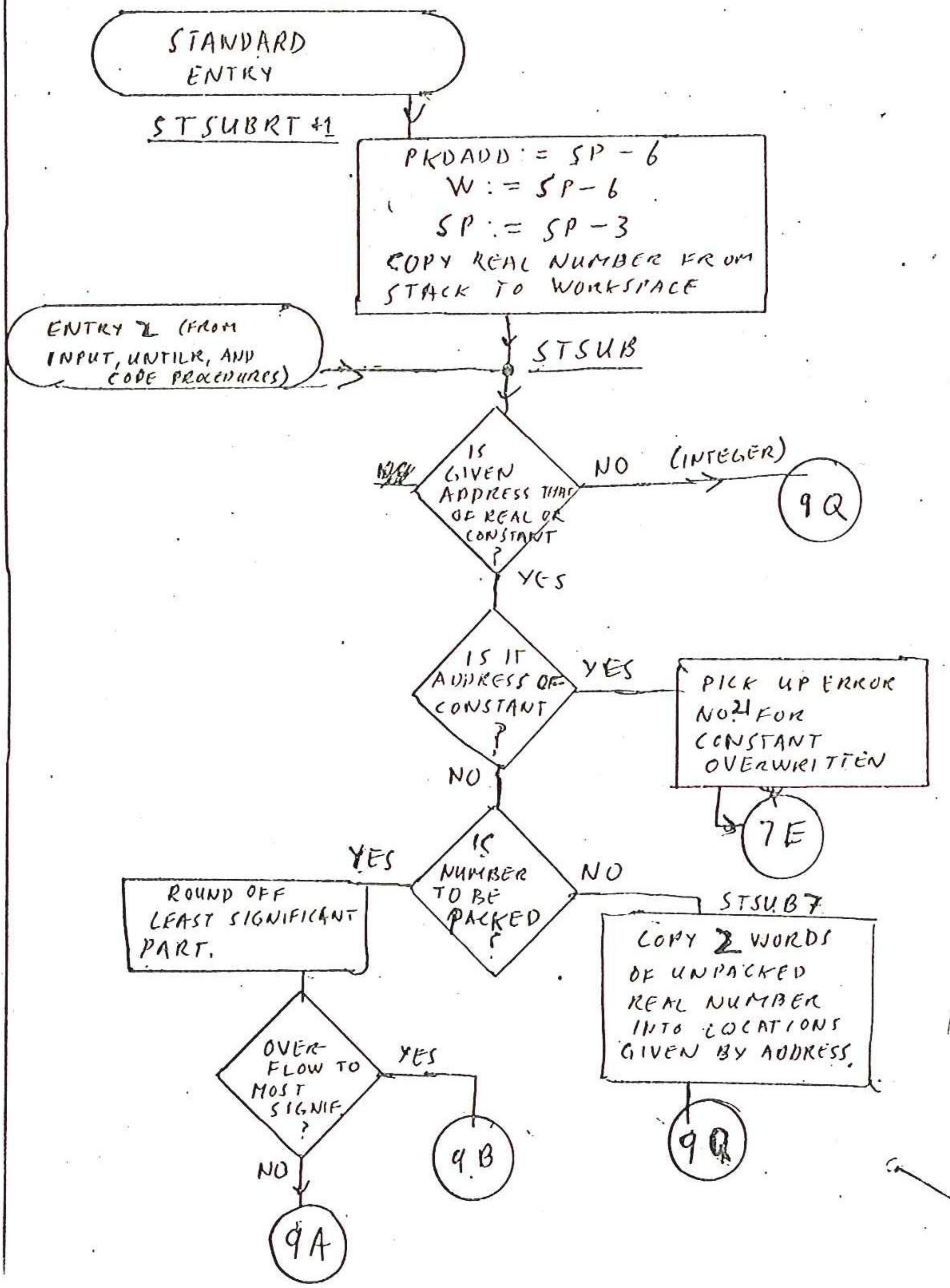

65
IS PRIMITIVE NUMBER < 57 ?
NO
W:=SP − 3
PRIMA
YES
W := SP − 6;
SP := SP − 3;
PRIMA1
PICK UP ADDRESS FROM PRIM TABLE. GO TO THE CORRESPONDING SUB-ROUTINE (WITH CONVENTIONAL ENTRY AND EXIT).
PRIM ~~SHEET~~ SHEET PRIM
30 (I+I→I)
31 (R+R→R)
32 (I−I→I)
33 R−R→R
34 I*I→I
35 R*R→R
36 I/I→R
37 R/R→R
38 I↑I→I
39 I↑I→R
40 R↑R→R
41 I<I→B
42 R<R→B
43 I≤I→B
44 R≤R→B
45 I=I→B
46 R=R→B
47 I≠I→B
48 R≠R→B
49 I>I→B
50 R>R→B
51 I≥I→B
52 R≥R→B
53 B∧B→B
54 B∨B→B
55 B≡B→B
56 B⊃B→B
57 ¬B→B
58 ABS(X)
59 ENTIER
60 EXP
61 LN
62 SIGN
63–70 Parameter checks
4N

FAIL Subroutine (FAILEN)

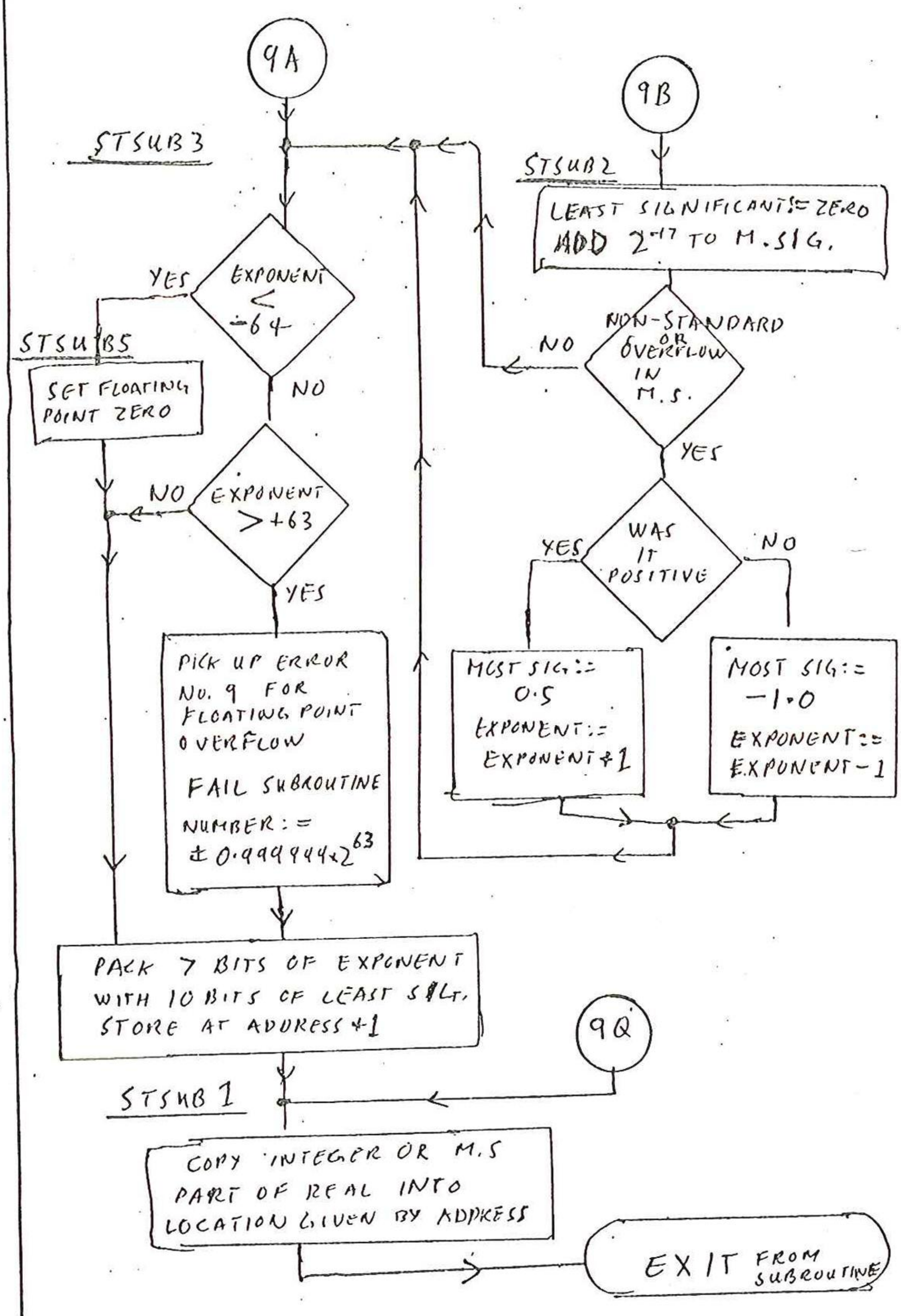


SUBRT LINK STORED IN PAUSRT
RE-ENTRY CAUSES CONTINUATION AT
NEXT INSTRUCTION OF USER PROGRAM.

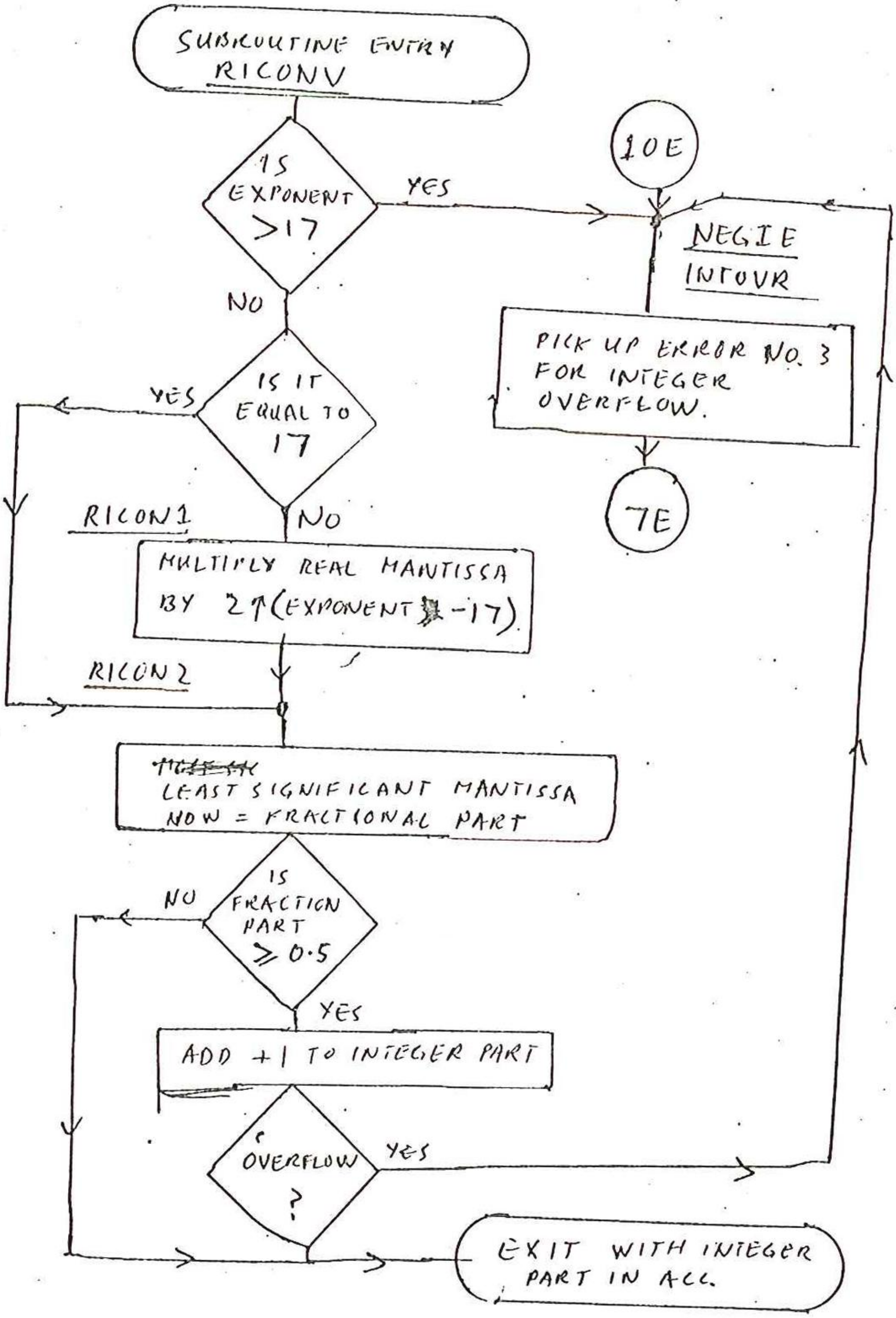
ASSIGNE Subroutine



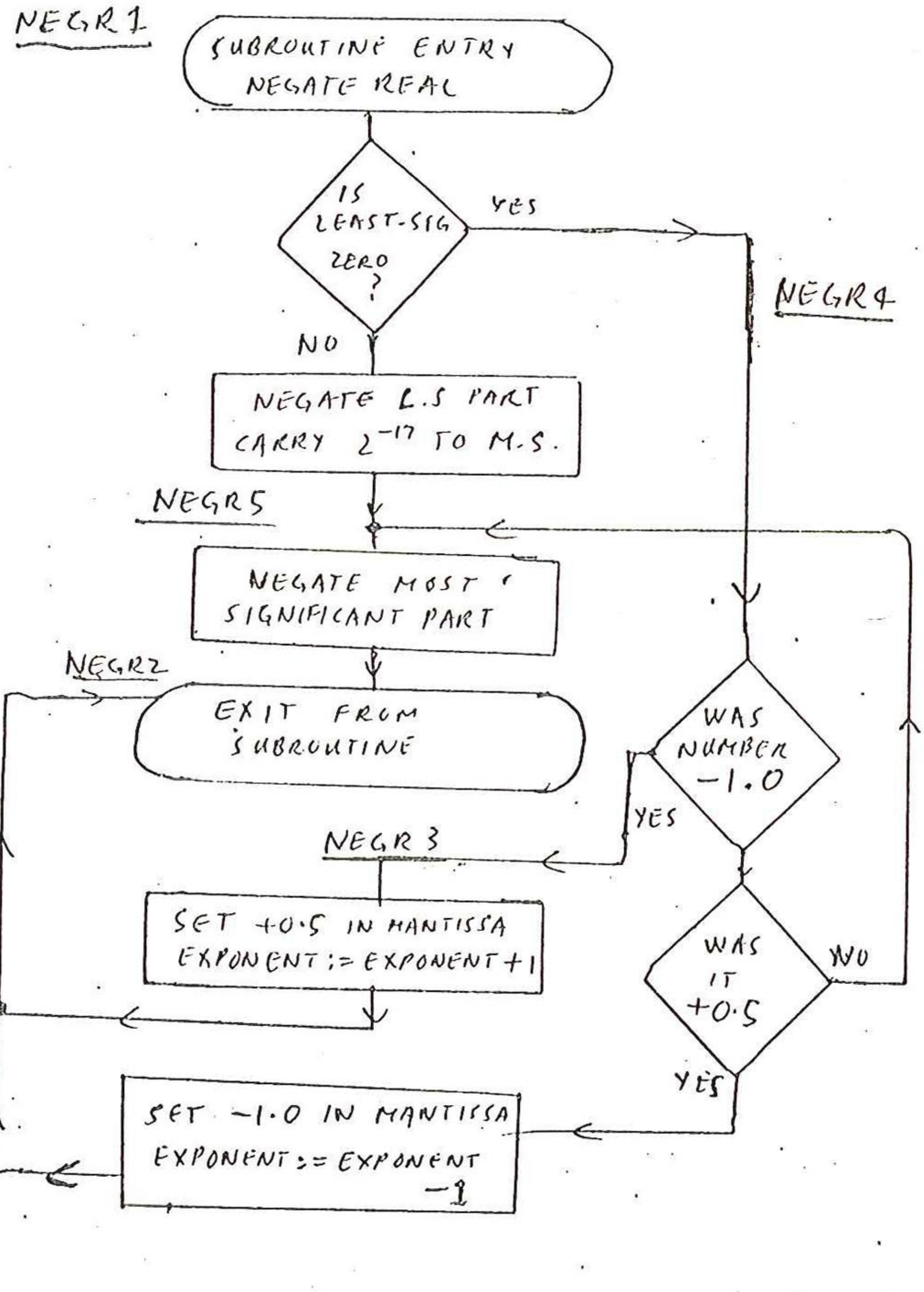
Sheet 9


9A
9B
STSUB3
STSUB2
LEAST SIGNIFICANT:= ZERO
ADD 2^{-17} TO M.SIG.
EXPONENT < -64
YES
NO
STSUB5
SET FLOATING POINT ZERO
NON-STANDARD OR OVERFLOW IN M.S.
NO
YES
EXPONENT > +63
NO
YES
WAS IT POSITIVE
YES
NO
PICK UP ERROR No. 9 FOR FLOATING POINT OVERFLOW
FAIL SUBROUTINE
NUMBER := $\pm 0.999999 \times 2^{63}$
MOST SIG:= 0.5
EXPONENT:= EXPONENT+1
MOST SIG:= -1.0
EXPONENT:= EXPONENT-1
PACK 7 BITS OF EXPONENT WITH 10 BITS OF LEAST SIGT. STORE AT ADDRESS+1
9Q
STSUB1
COPY INTEGER OR M.S PART OF REAL INTO LOCATION GIVEN BY ADDRESS
EXIT FROM SUBROUTINE

REAL TO INTEGER SUBROUTINE

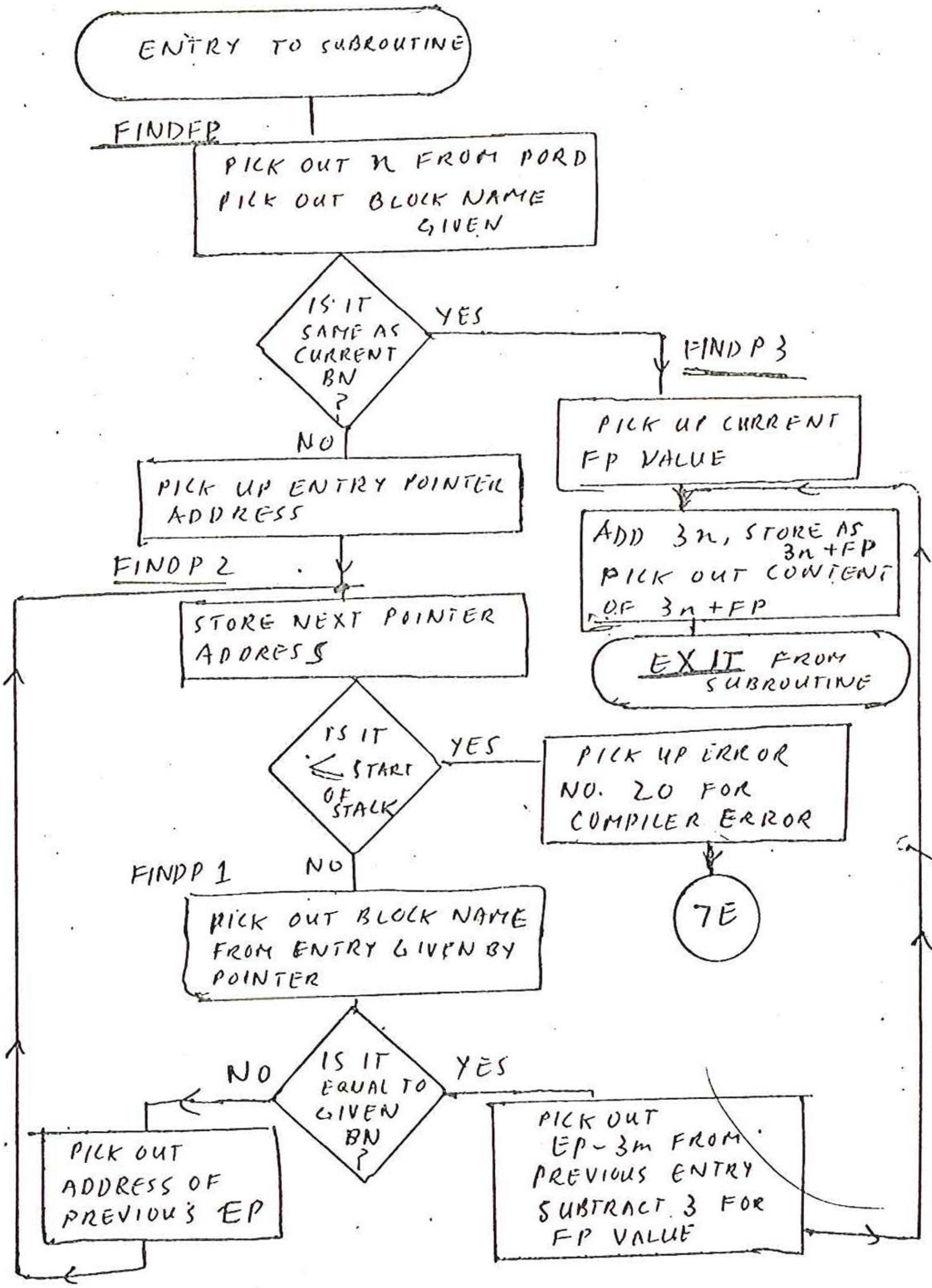


Sheet 11



NEGR1
SUBROUTINE ENTRY NEGATE REAL
IS LEAST-SIG ZERO ?
YES
NO
NEGR4
NEGATE L.S PART CARRY 2^{-17} TO M.S.
NEGR5
NEGATE MOST SIGNIFICANT PART
NEGR2
EXIT FROM SUBROUTINE
WAS NUMBER −1.0
YES
NEGR3
SET +0.5 IN MANTISSA EXPONENT := EXPONENT+1
WAS IT +0.5
NO
YES
SET −1.0 IN MANTISSA EXPONENT := EXPONENT −1

SUBROUTINE: FIND FORMAL POINTER AND PARAMETER POSITION



Function PEM (13 P)

n := ADPART
STORE EP − 3n at SP
FP := EP − 3n − 3
Store BN at SP+1
SP := SP+2

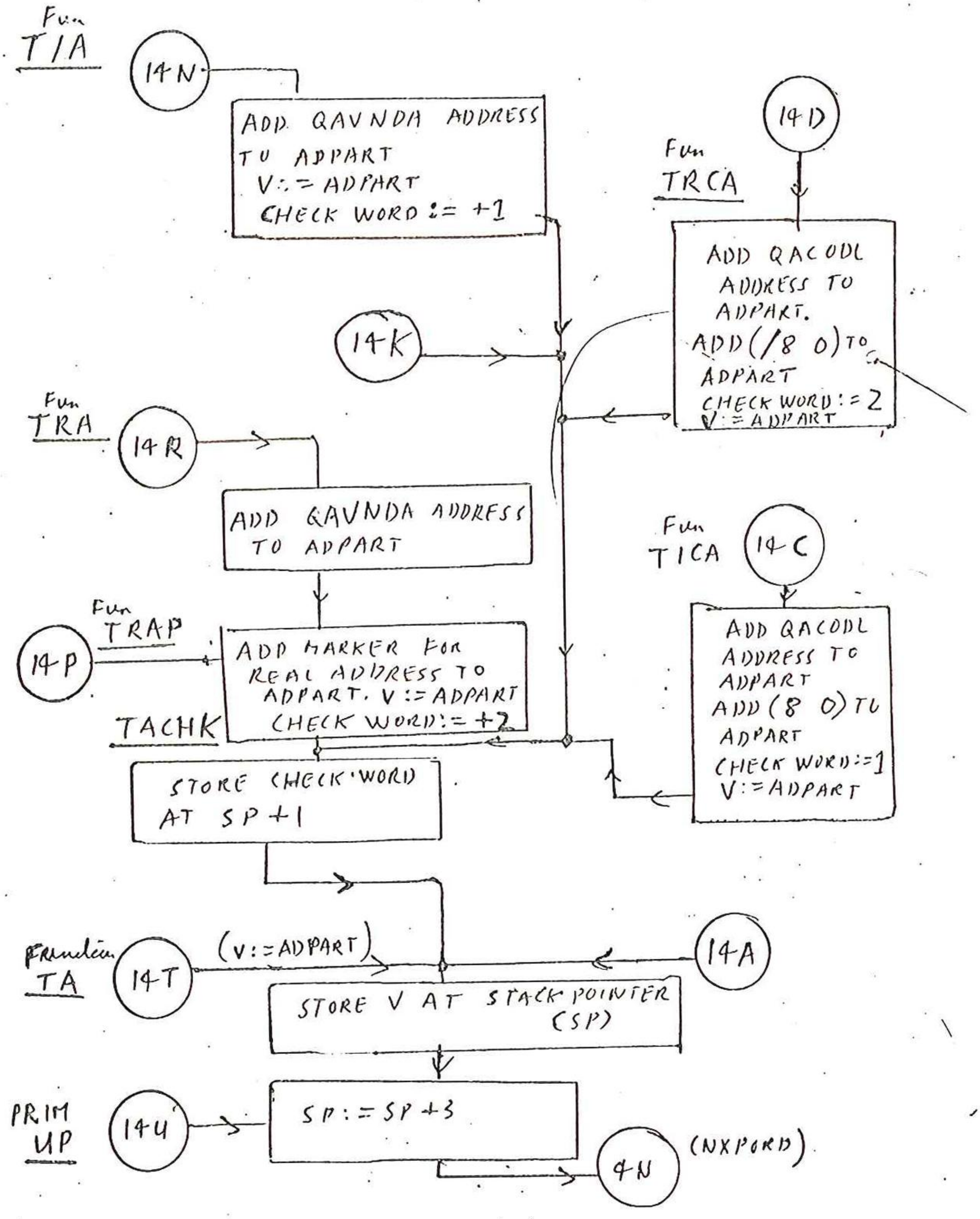
CALL MACHINE CODE SUBROUTINE
(LINK AT (PP), ENTRY AT ((PP)+1)

PRIM RETURN (13R)

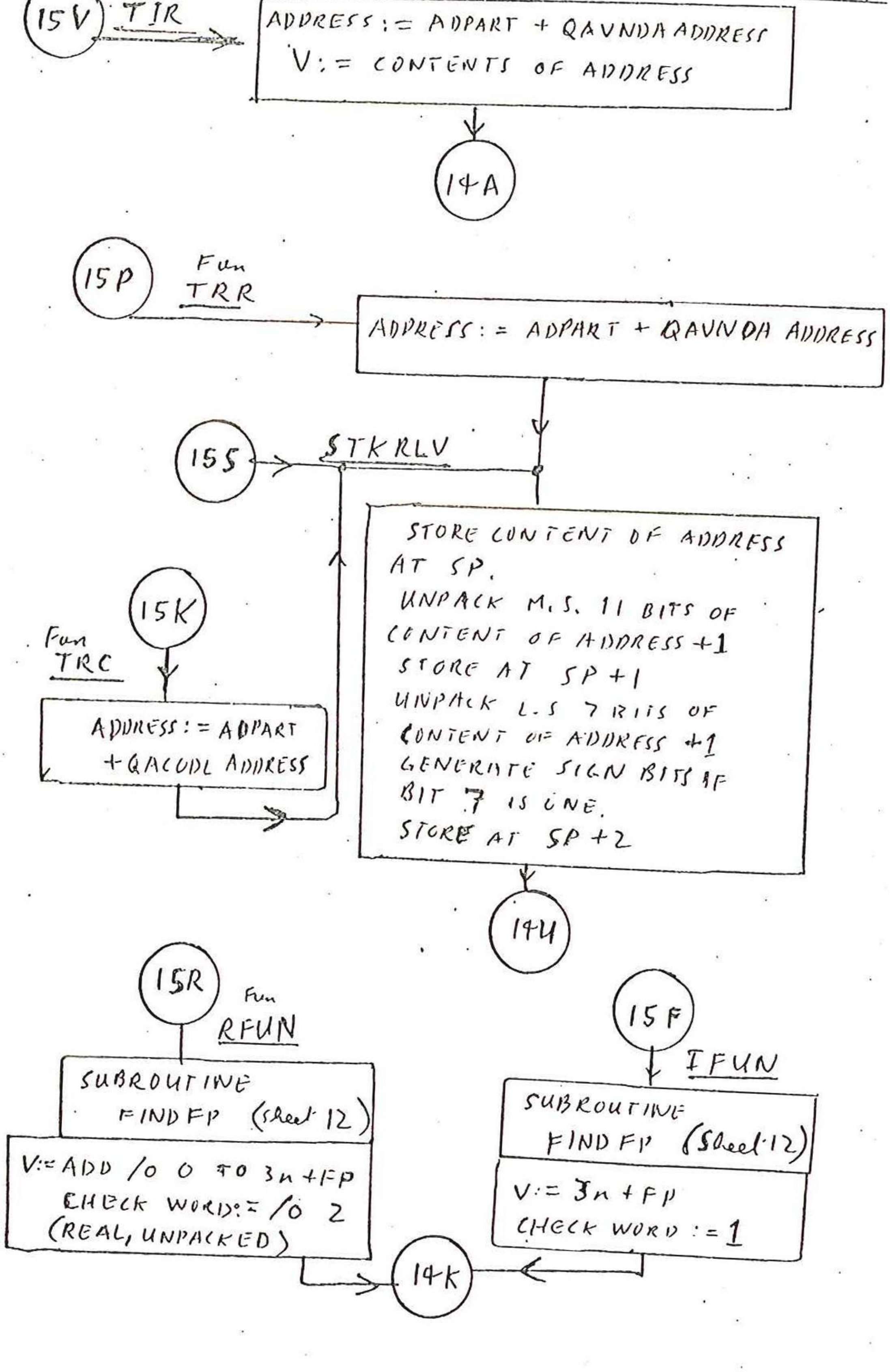
BN := CONTENT OF EP+3
SP := CONTENT OF EP+2
PP := CONTENT OF EP+1
EP := CONTENT OF EP
~~FP := (CONTENT OF EP+2) − 3~~

(4N) (NXPORD)

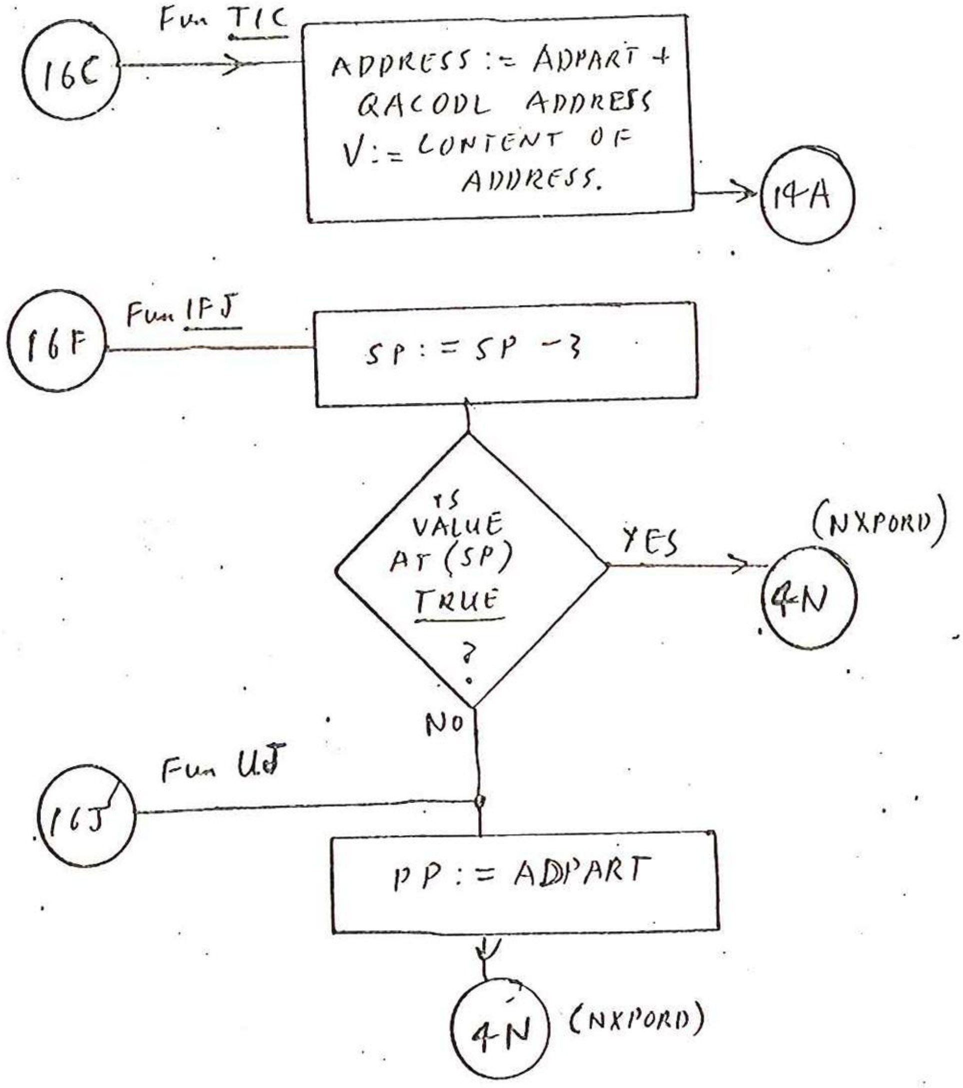
Sheet 14


Fun T/A
14N
ADD QAVNDA ADDRESS TO ADPART V:=ADPART CHECK WORD := +1
14D
Fun TRCA
ADD QACODE ADDRESS TO ADPART. ADD(/8 0) TO ADPART CHECK WORD := 2 V:=ADPART
14K
Fun TRA
14R
ADD QAVNDA ADDRESS TO ADPART
Fun TICA
14C
Fun TRAP
14P
ADD MARKER FOR REAL ADDRESS TO ADPART. V:=ADPART CHECK WORD:= +2
TACHK
ADD QACODE ADDRESS TO ADPART ADD (8 0) TO ADPART CHECK WORD:=1 V:=ADPART
STORE CHECK WORD AT SP+1
Function TA
14T
(V:=ADPART)
14A
STORE V AT STACK POINTER (SP)
PRIM UP
14U
SP:=SP+3
4N
(NXPORD)

Sheet 15

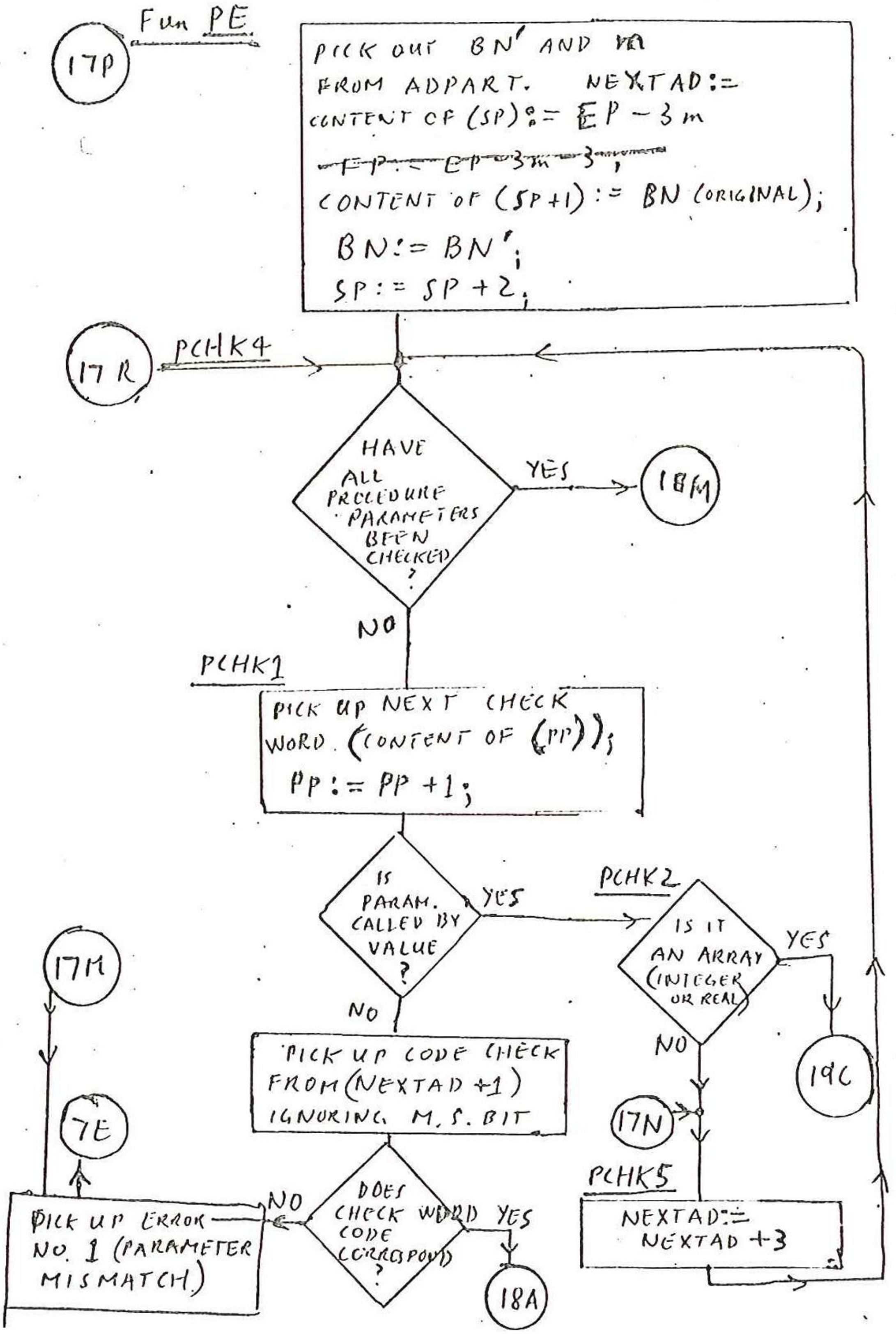

15V
Fun
TIR
ADDRESS := ADPART + QAVNDA ADDRESS
V := CONTENTS OF ADDRESS
14A
15P
Fun
TRR
ADDRESS := ADPART + QAVNDA ADDRESS
15S
STKRLV
STORE CONTENT OF ADDRESS AT SP.
UNPACK M.S. 11 BITS OF CONTENT OF ADDRESS +1
STORE AT SP+1
UNPACK L.S 7 BITS OF CONTENT OF ADDRESS +1
GENERATE SIGN BITS IF BIT 7 IS ONE.
STORE AT SP+2
15K
Fun
TRC
ADDRESS := ADPART + QACODL ADDRESS
14U
15R
Fun
RFUN
SUBROUTINE FINDFP (Sheet 12)
V:= ADD /0 0 TO 3n+FP
CHECK WORD:= /0 2
(REAL, UNPACKED)
15F
IFUN
SUBROUTINE FINDFP (Sheet 12)
V:= 3n+FP
CHECK WORD := 1
14K

Sheet 16



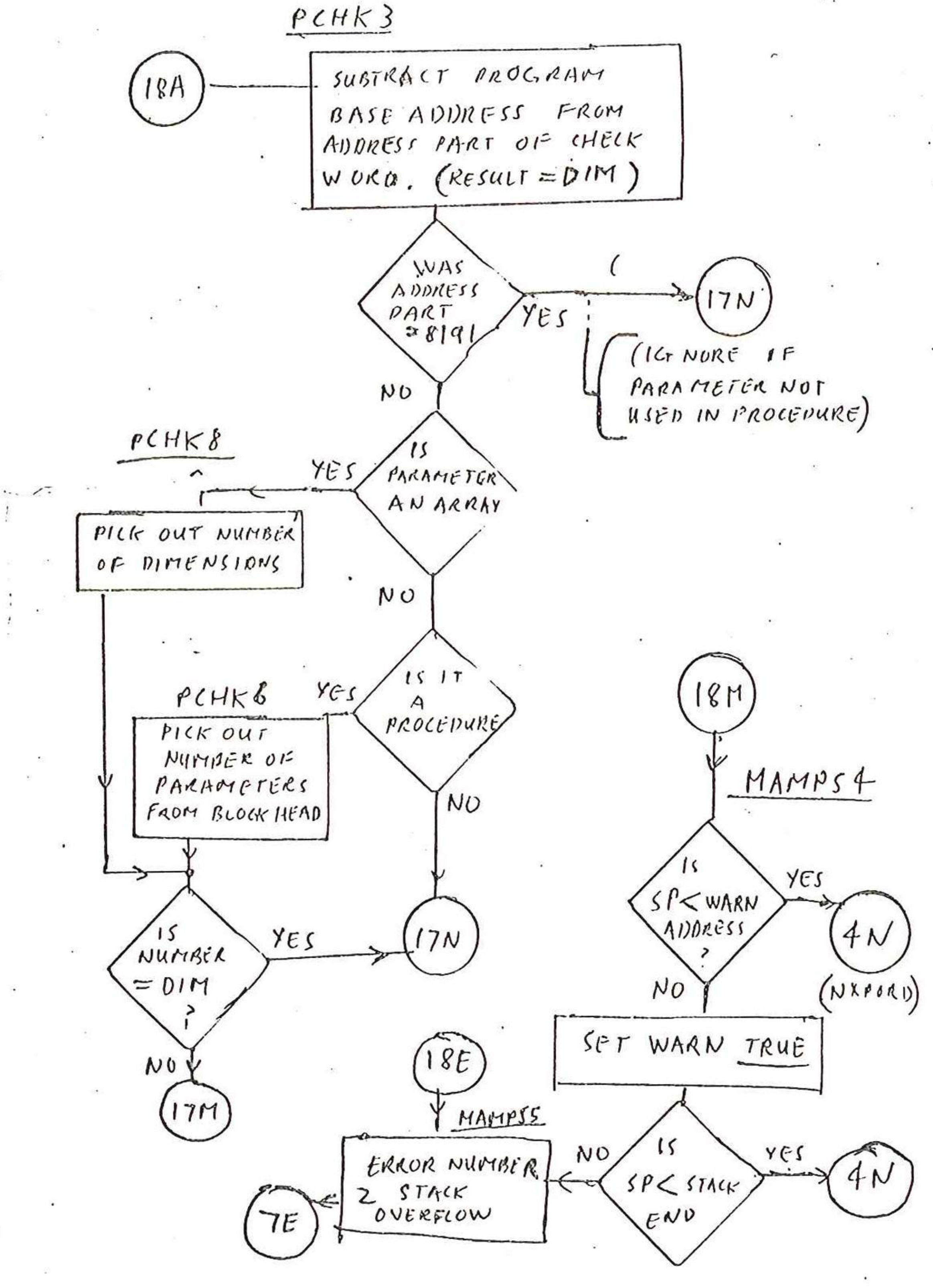
16C
Fun TIC
ADDRESS := ADPART + QACODL ADDRESS
V := CONTENT OF ADDRESS.
14A
16F
Fun IFJ
SP := SP - 3
IS VALUE AT (SP) TRUE ?
YES
(NXPORD)
4N
NO
16J
Fun UJ
PP := ADPART
4N
(NXPORD)

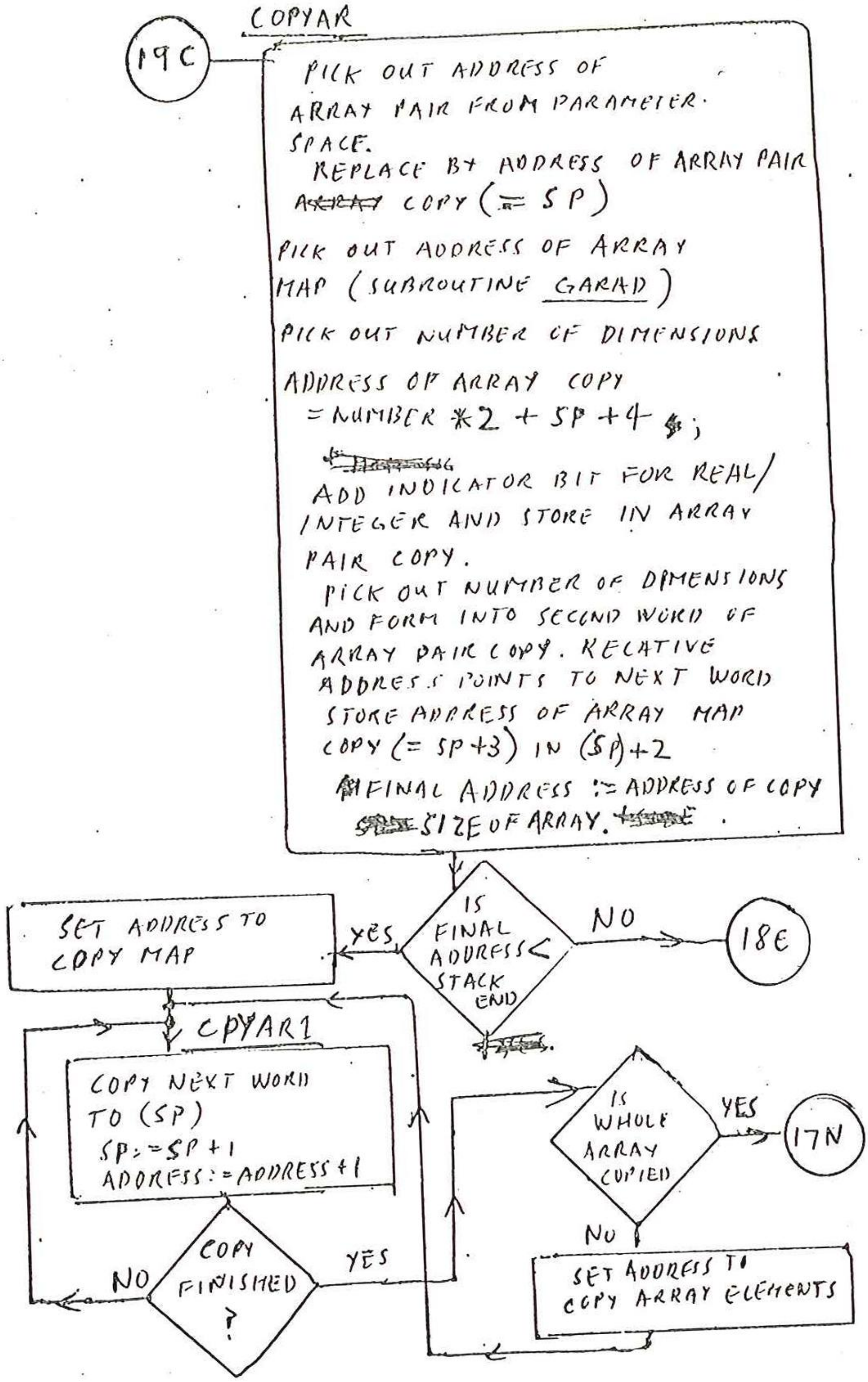
Sheet 17



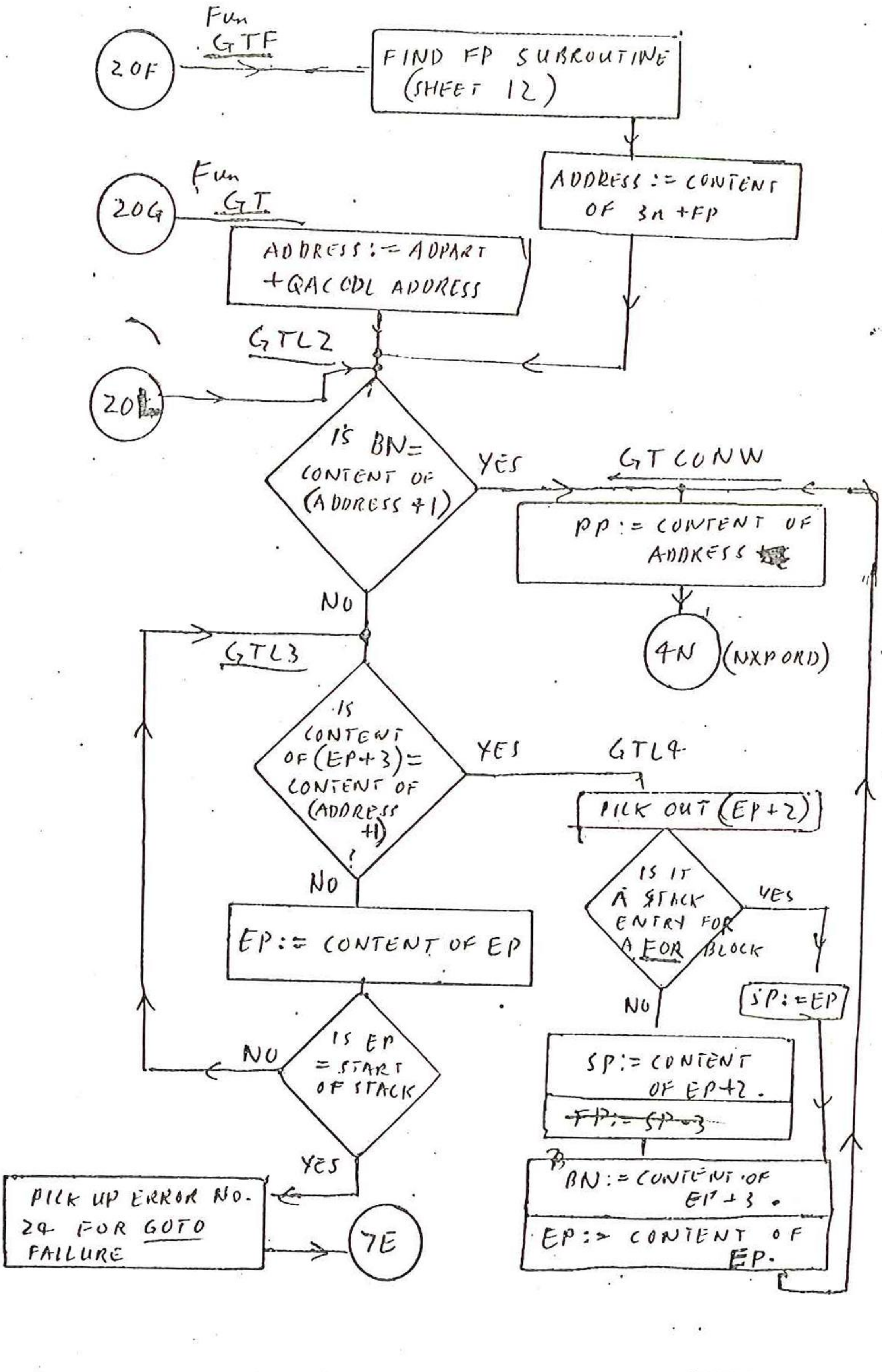
Fun PE
17P
PICK OUT BN' AND m
FROM ADPART. NEXTAD:=
CONTENT OF (SP):= EP - 3m
~~EP:= EP - 3m - 3;~~
CONTENT OF (SP+1):= BN (ORIGINAL);
BN:= BN';
SP:= SP + 2;
17R
PCHK4
HAVE ALL PROCEDURE PARAMETERS BEEN CHECKED ?
YES
18M
NO
PCHK1
PICK UP NEXT CHECK WORD. (CONTENT OF (PP));
PP:= PP + 1;
IS PARAM. CALLED BY VALUE ?
YES
PCHK2
IS IT AN ARRAY (INTEGER OR REAL)
YES
19C
NO
17N
PCHK5
NEXTAD:= NEXTAD + 3
NO
PICK UP CODE CHECK FROM (NEXTAD + 1) IGNORING M.S. BIT
DOES CHECK WORD CODE CORRESPOND ?
YES
18A
NO
PICK UP ERROR NO. 1 (PARAMETER MISMATCH)
17M
7E

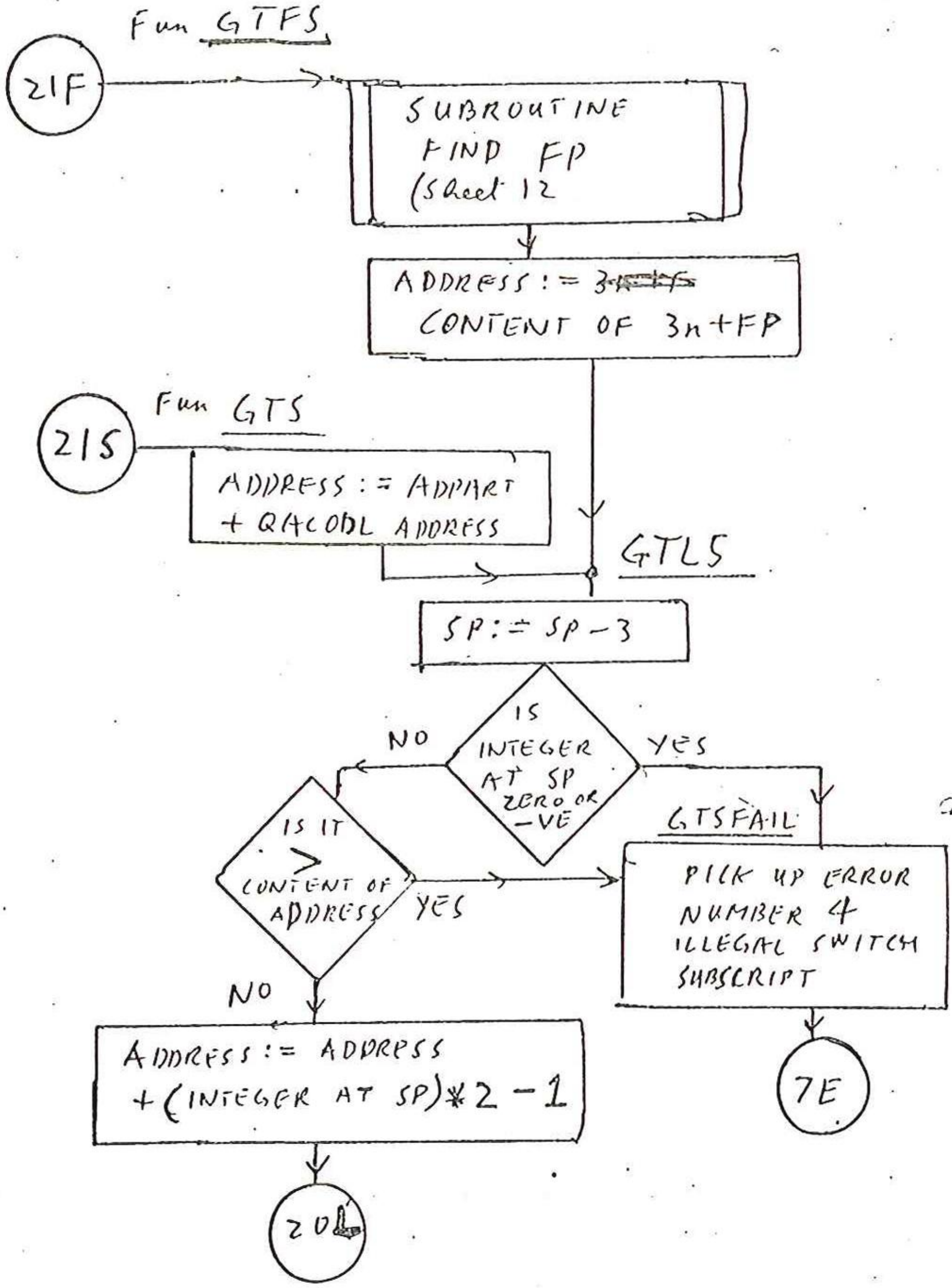
Sheet 18


PCHK3
18A
SUBTRACT PROGRAM BASE ADDRESS FROM ADDRESS PART OF CHECK WORD. (RESULT = DIM)
WAS ADDRESS PART = 8191
YES
17N
(IGNORE IF PARAMETER NOT USED IN PROCEDURE)
NO
PCHK8
IS PARAMETER AN ARRAY
YES
PICK OUT NUMBER OF DIMENSIONS
NO
IS IT A PROCEDURE
YES
PCHK6
PICK OUT NUMBER OF PARAMETERS FROM BLOCK HEAD
NO
17N
IS NUMBER = DIM ?
YES
NO
17M
18M
MAMPS4
IS SP < WARN ADDRESS ?
YES
4N
(NXPORD)
NO
SET WARN TRUE
18E
MAMPS5
IS SP < STACK END
NO
YES
4N
ERROR NUMBER 2 STACK OVERFLOW
7E

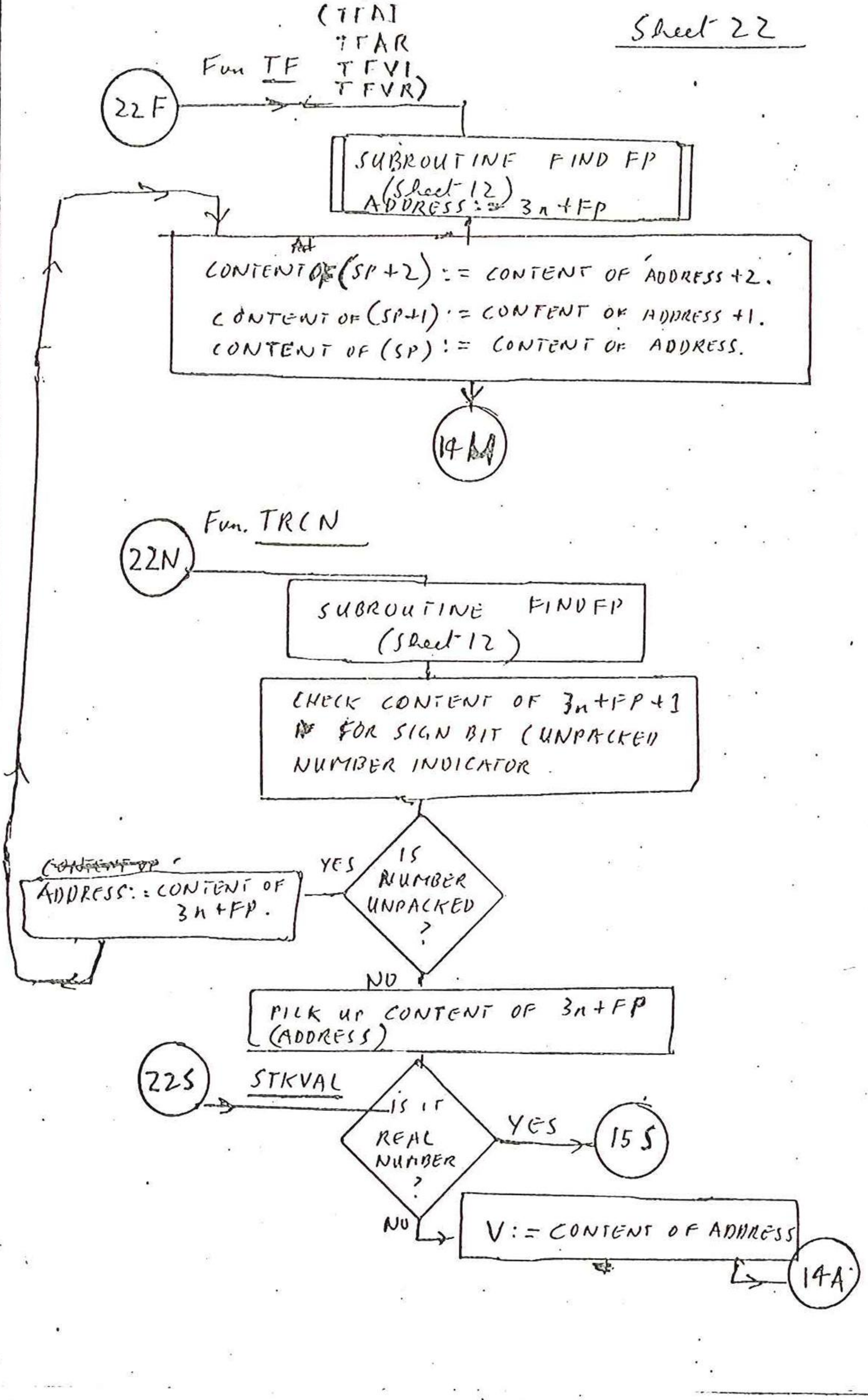

COPYAR
19C
PICK OUT ADDRESS OF ARRAY PAIR FROM PARAMETER SPACE.
REPLACE BY ADDRESS OF ARRAY PAIR ~~ARRAY~~ COPY (= SP)
PICK OUT ADDRESS OF ARRAY MAP (SUBROUTINE GARAD)
PICK OUT NUMBER OF DIMENSIONS
ADDRESS OF ARRAY COPY = NUMBER *2 + SP + 4 ;
ADD INDICATOR BIT FOR REAL/INTEGER AND STORE IN ARRAY PAIR COPY.
PICK OUT NUMBER OF DIMENSIONS AND FORM INTO SECOND WORD OF ARRAY PAIR COPY. RELATIVE ADDRESS POINTS TO NEXT WORD
STORE ADDRESS OF ARRAY MAP COPY (= SP+3) IN (SP)+2
FINAL ADDRESS := ADDRESS OF COPY SIZE OF ARRAY.
IS FINAL ADDRESS < STACK END
NO
18E
YES
SET ADDRESS TO COPY MAP
CPYAR1
COPY NEXT WORD TO (SP)
SP:=SP+1
ADDRESS:=ADDRESS+1
COPY FINISHED ?
NO
YES
IS WHOLE ARRAY COPIED
YES
17N
NO
SET ADDRESS TO COPY ARRAY ELEMENTS

Sheet 20

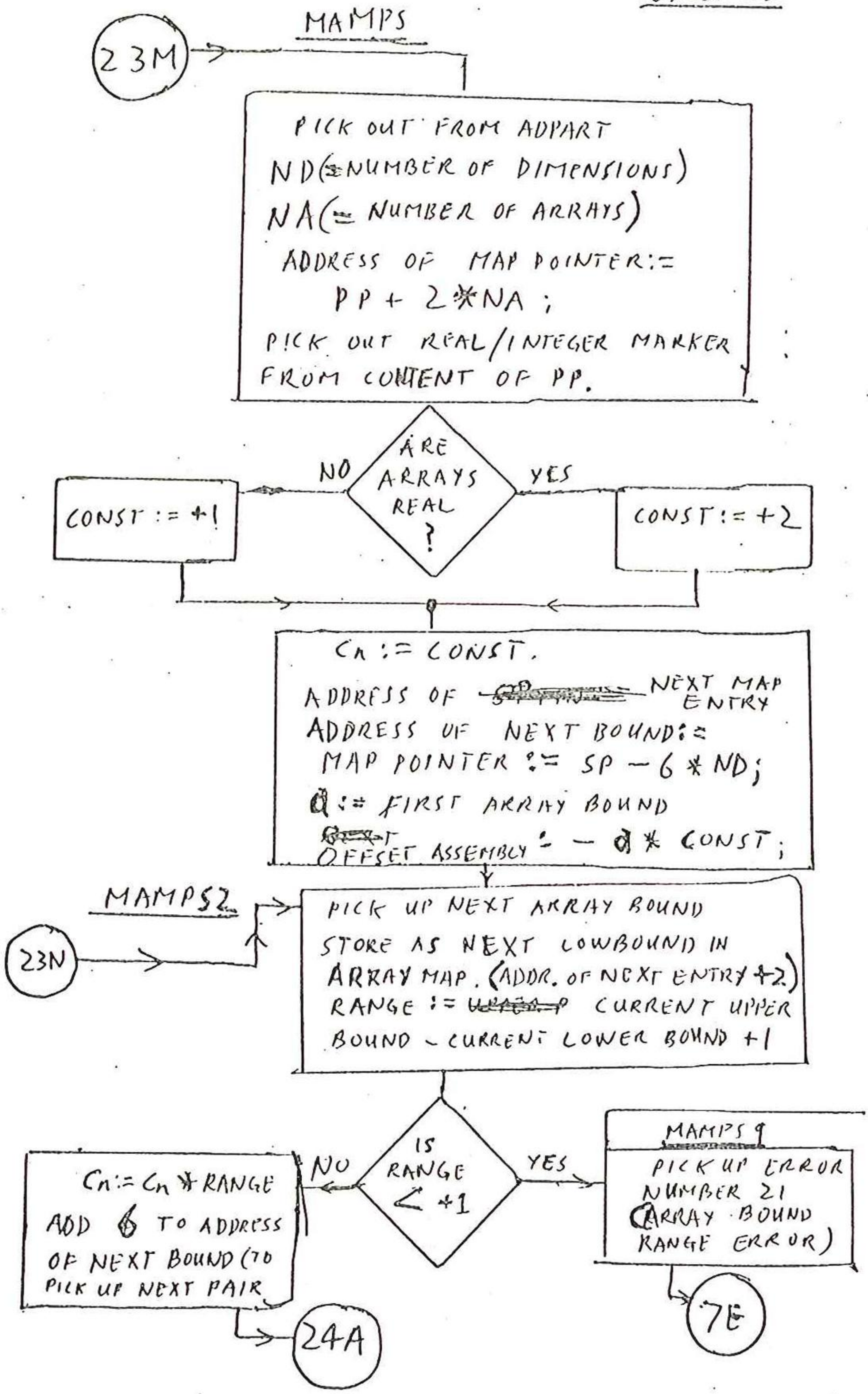

20F
Fun GTF
FIND FP SUBROUTINE (SHEET 12)
ADDRESS := CONTENT OF 3n + FP
20G
Fun GT
ADDRESS := ADPART + QACODL ADDRESS
GTL2
20L
IS BN = CONTENT OF (ADDRESS + 1)
YES
GTCONW
PP := CONTENT OF ADDRESS
4N
(NXPORD)
NO
GTL3
IS CONTENT OF (EP+3) = CONTENT OF (ADDRESS +1)
YES
GTL4
PICK OUT (EP+2)
IS IT A STACK ENTRY FOR A FOR BLOCK
YES
SP:=EP
NO
NO
EP := CONTENT OF EP
IS EP = START OF STACK
NO
SP := CONTENT OF EP+2.
~~FP := SP+3~~
YES
BN := CONTENT OF EP+3.
PICK UP ERROR NO. 24 FOR GOTO FAILURE
7E
EP := CONTENT OF EP.


Fun GTFS
21F
SUBROUTINE
FIND FP
(Sheet 12
ADDRESS := ~~3n+FP~~
CONTENT OF 3n+FP
Fun GTS
21S
ADDRESS := ADPART
+ QACODL ADDRESS
GTL5
SP := SP − 3
IS
INTEGER
AT SP
ZERO OR
−VE
NO
YES
IS IT
>
CONTENT OF
ADDRESS
YES
NO
GTSFAIL
PICK UP ERROR
NUMBER 4
ILLEGAL SWITCH
SUBSCRIPT
7E
ADDRESS := ADDRESS
+ (INTEGER AT SP)*2 − 1
20L

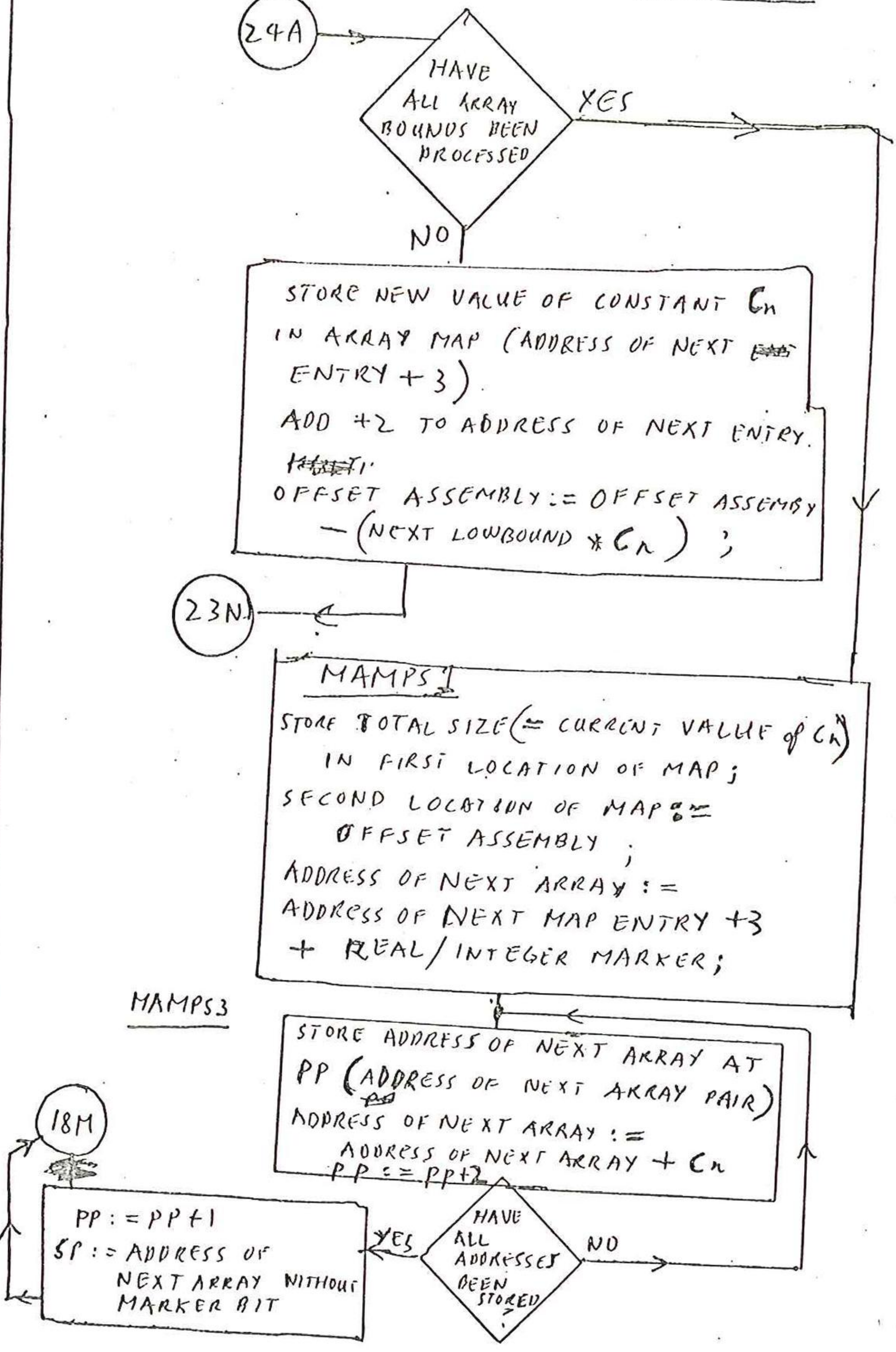
Sheet 22


Fun TF
(TFAI
TFAR
TFVI
TFVR)
22F
SUBROUTINE FIND FP
(Sheet 12)
ADDRESS := 3n+FP
CONTENT OF (SP+2) := CONTENT OF ADDRESS+2.
CONTENT OF (SP+1) := CONTENT OF ADDRESS+1.
CONTENT OF (SP) := CONTENT OF ADDRESS.
14M
Fun. TRCN
22N
SUBROUTINE FINDFP
(Sheet 12)
CHECK CONTENT OF 3n+FP+1
FOR SIGN BIT (UNPACKED
NUMBER INDICATOR
IS NUMBER UNPACKED ?
YES
ADDRESS := CONTENT OF 3n+FP.
NO
PICK UP CONTENT OF 3n+FP
(ADDRESS)
22S
STKVAL
IS IT REAL NUMBER ?
YES
15S
NO
V := CONTENT OF ADDRESS
14A

Sheet 23


MAMPS
23M
PICK OUT FROM ADPART
ND(= NUMBER OF DIMENSIONS)
NA(= NUMBER OF ARRAYS)
ADDRESS OF MAP POINTER := PP + 2*NA ;
PICK OUT REAL/INTEGER MARKER FROM CONTENT OF PP.
ARE ARRAYS REAL ?
NO
YES
CONST := +1
CONST := +2
Cn := CONST.
ADDRESS OF NEXT MAP ENTRY
ADDRESS OF NEXT BOUND :=
MAP POINTER := SP − 6 * ND;
d := FIRST ARRAY BOUND
OFFSET ASSEMBLY := − d * CONST;
MAMPS2
23N
PICK UP NEXT ARRAY BOUND
STORE AS NEXT LOWBOUND IN ARRAY MAP. (ADDR. OF NEXT ENTRY +2)
RANGE := CURRENT UPPER BOUND − CURRENT LOWER BOUND +1
IS RANGE < +1
NO
YES
Cn := Cn * RANGE
ADD 6 TO ADDRESS OF NEXT BOUND (TO PICK UP NEXT PAIR
24A
MAMPS 9
PICK UP ERROR NUMBER 21 (ARRAY BOUND RANGE ERROR)
7E

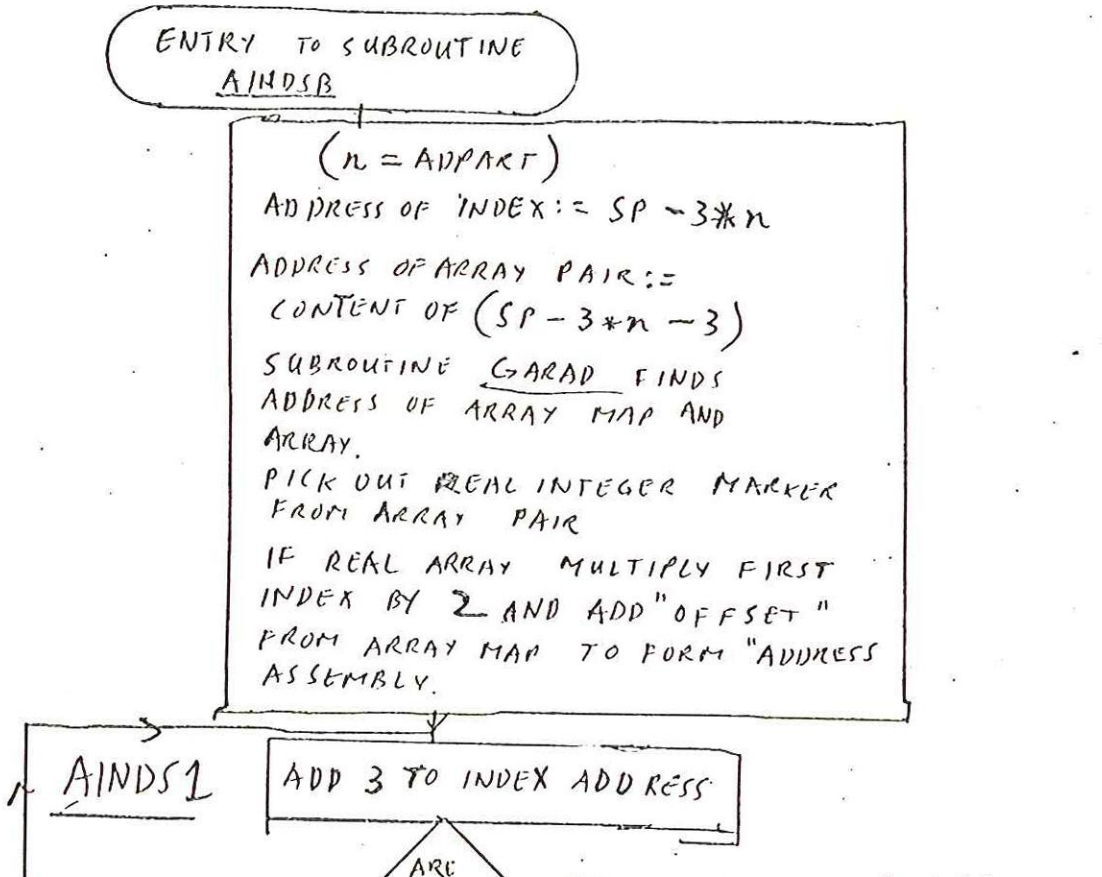
Sheet 24



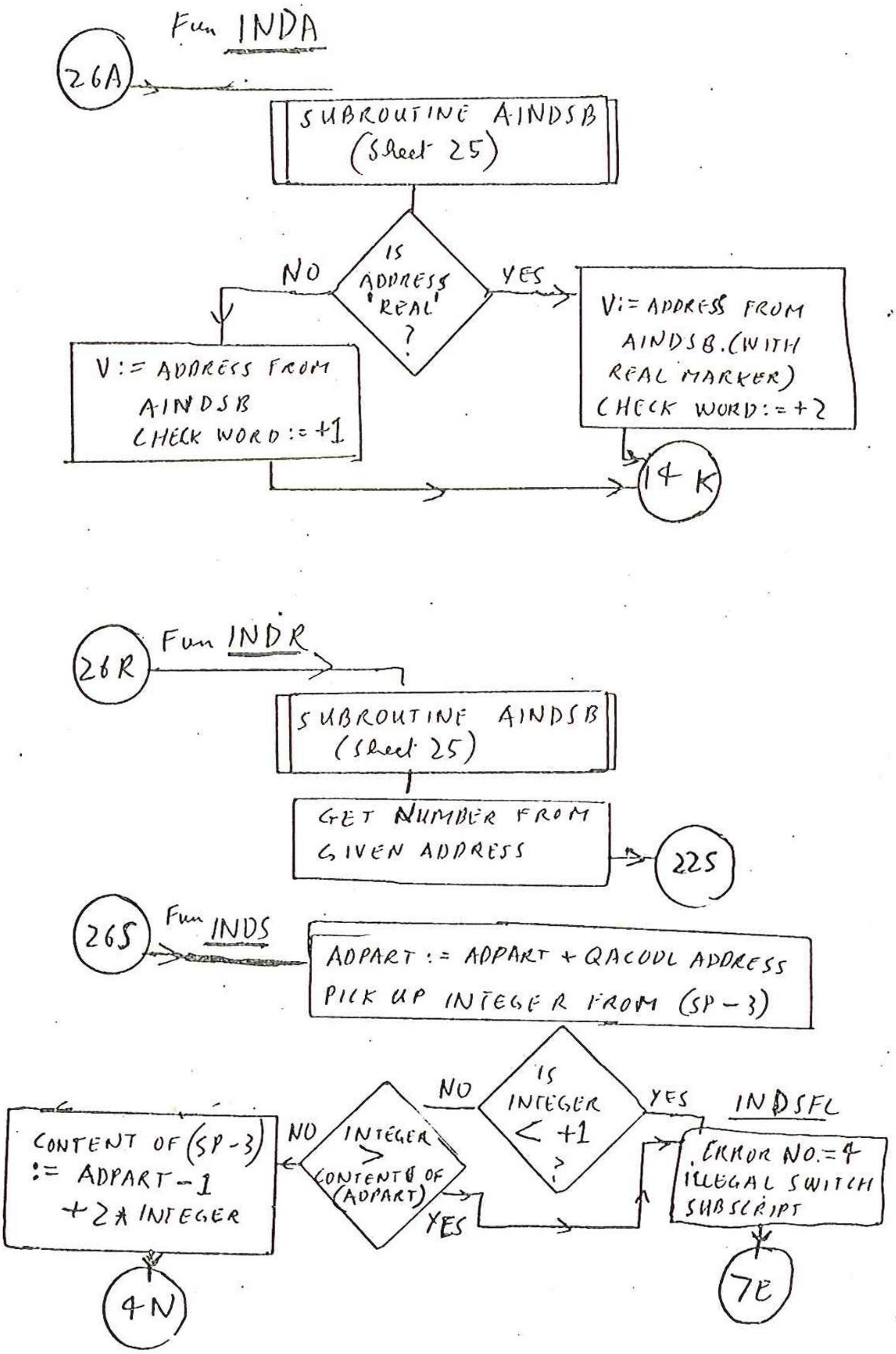
24A
HAVE ALL ARRAY BOUNDS BEEN PROCESSED
YES
NO
STORE NEW VALUE OF CONSTANT C_n IN ARRAY MAP (ADDRESS OF NEXT ENTRY + 3).
ADD +2 TO ADDRESS OF NEXT ENTRY.
OFFSET ASSEMBLY := OFFSET ASSEMBLY − (NEXT LOWBOUND * C_n) ;
23N
MAMPS1
STORE TOTAL SIZE (= CURRENT VALUE of C_n) IN FIRST LOCATION OF MAP;
SECOND LOCATION OF MAP := OFFSET ASSEMBLY ;
ADDRESS OF NEXT ARRAY := ADDRESS OF NEXT MAP ENTRY +3 + REAL/INTEGER MARKER;
MAMPS3
STORE ADDRESS OF NEXT ARRAY AT PP (ADDRESS OF NEXT ARRAY PAIR)
ADDRESS OF NEXT ARRAY := ADDRESS OF NEXT ARRAY + C_n
PP := PP+2
18M
PP := PP+1
SP := ADDRESS OF NEXT ARRAY WITHOUT MARKER BIT
YES
HAVE ALL ADDRESSES BEEN STORED
NO

Sheet 25

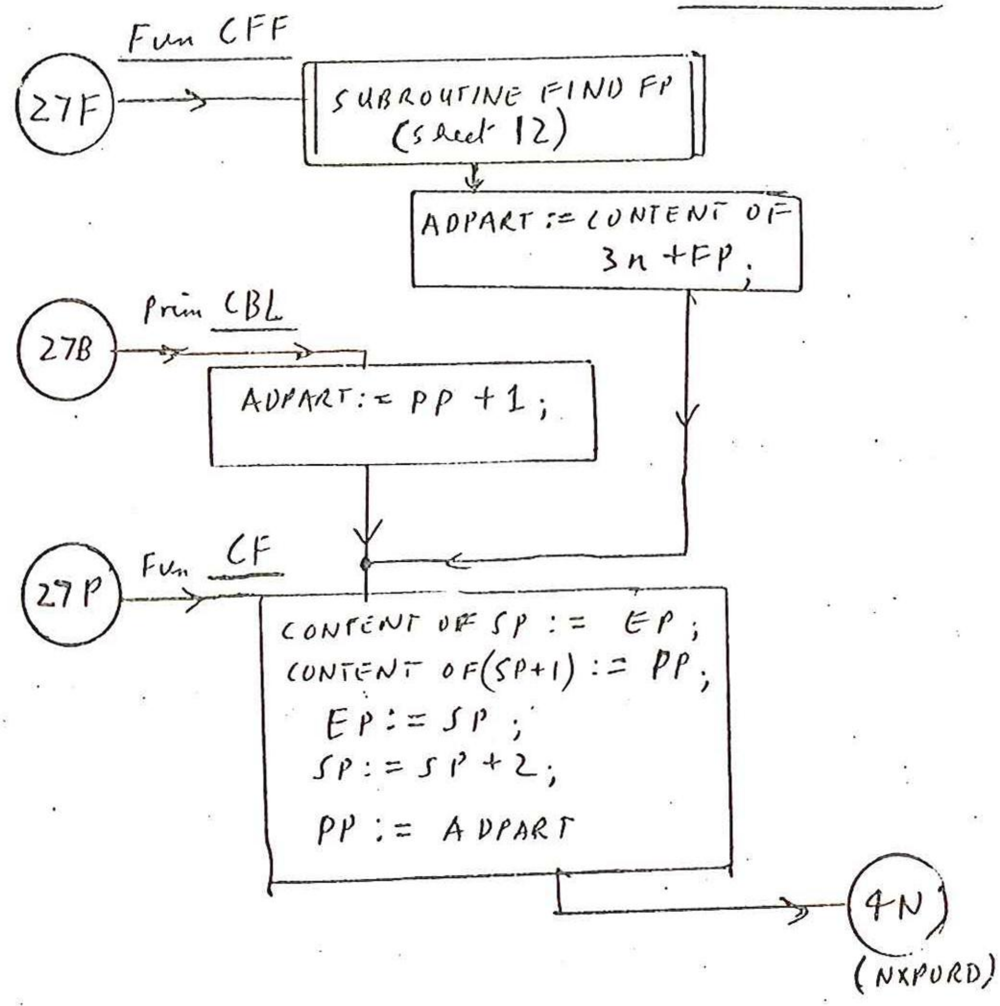
SUBROUTINE: AINDSB



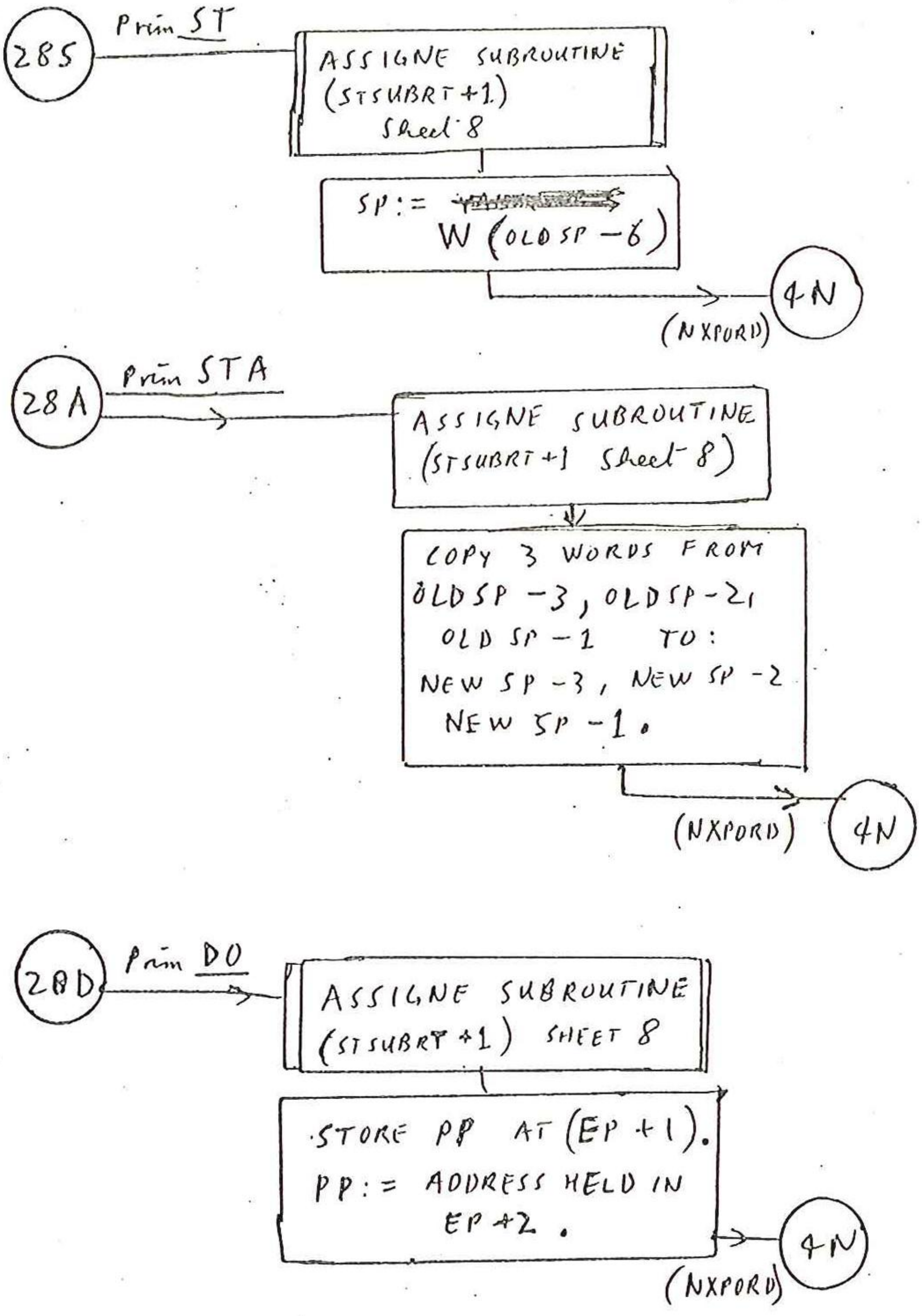
Sheet 26

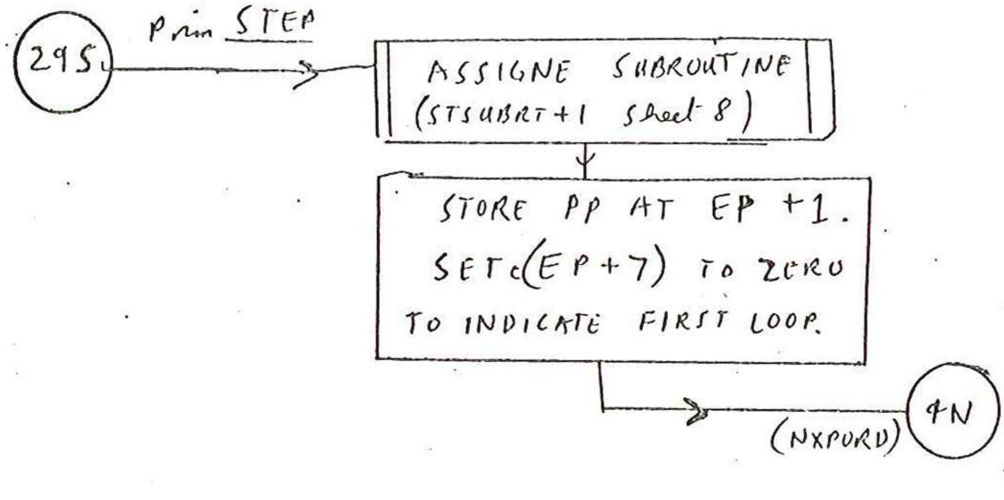

Fun INDA
26A
SUBROUTINE AINDSB (Sheet 25)
IS ADDRESS 'REAL' ?
NO
YES
V := ADDRESS FROM AINDSB CHECK WORD := +1
V := ADDRESS FROM AINDSB. (WITH REAL MARKER) CHECK WORD := +2
14K
Fun INDR
26R
SUBROUTINE AINDSB (Sheet 25)
GET NUMBER FROM GIVEN ADDRESS
22S
Fun INDS
26S
ADPART := ADPART + QACODL ADDRESS
PICK UP INTEGER FROM (SP - 3)
IS INTEGER < +1 ?
NO
YES
INDSFL
ERROR NO. = 4 ILLEGAL SWITCH SUBSCRIPT
7E
INTEGER > CONTENTS OF (ADPART)
NO
YES
CONTENT OF (SP - 3) := ADPART - 1 + 2 * INTEGER
4N

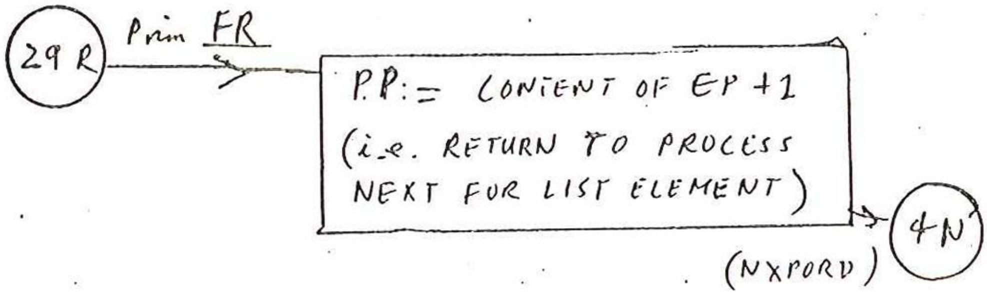
Sheet 27

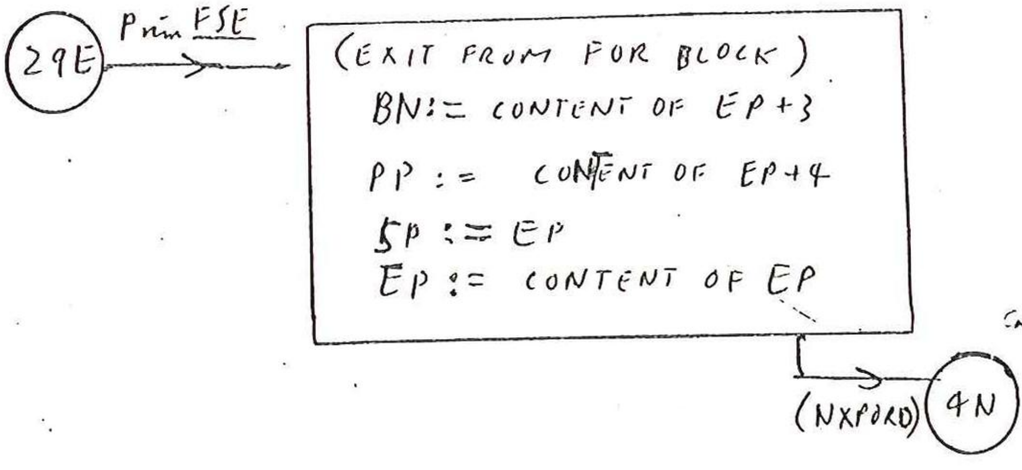

Fun CFF
27F
SUBROUTINE FIND FP
(Sheet 12)
ADPART := CONTENT OF 3n + FP;
Prim CBL
27B
ADPART := PP + 1;
Fun CF
27P
CONTENT OF SP := EP;
CONTENT OF (SP+1) := PP;
EP := SP;
SP := SP + 2;
PP := ADPART
4N
(NXPORD)

Sheet 28


28S
Prim ST
ASSIGNE SUBROUTINE
(STSUBRT +1)
Sheet 8
SP := W (OLDSP −6)
4N
(NXPORD)
28A
Prim STA
ASSIGNE SUBROUTINE
(STSUBRT +1 Sheet 8)
COPY 3 WORDS FROM
OLDSP −3, OLDSP −2,
OLD SP −1 TO:
NEW SP −3, NEW SP −2
NEW SP −1.
(NXPORD)
4N
28D
Prim DO
ASSIGNE SUBROUTINE
(STSUBRT +1) SHEET 8
STORE PP AT (EP +1).
PP := ADDRESS HELD IN
EP +2.
4N
(NXPORD)


29S
Prim STEP
ASSIGNE SUBROUTINE (STSUBRT+1 Sheet 8)
STORE PP AT EP +1. SET (EP+7) TO ZERO TO INDICATE FIRST LOOP.
(NXPORD)
4N


29R
Prim FR
P.P := CONTENT OF EP +1
(i.e. RETURN TO PROCESS NEXT FOR LIST ELEMENT)
(NXPORD)
4N


29E
Prim FSE
(EXIT FROM FOR BLOCK)
BN := CONTENT OF EP+3
PP := CONTENT OF EP+4
SP := EP
EP := CONTENT OF EP
(NXPORD)
4N

30F — Prim FOR

SET UP FOR BLOCK TO CONTROL FOR LOOP.
ADDRESS OF BLOCK = SP
c(BLOCK) := EP.
c(BLOCK+2) := ADDRESS 1 (= CONTENT OF PP).
c(BLOCK+3) := BN;
BN := CONTENT OF PP+1
CONTENT OF (BLOCK+4) := ADDRESS 2
(= CONTENT OF PP+2)
EP := SP
SP := SP+5; PP := PP+3

(NXPORD) 4N

30S — Prim STW

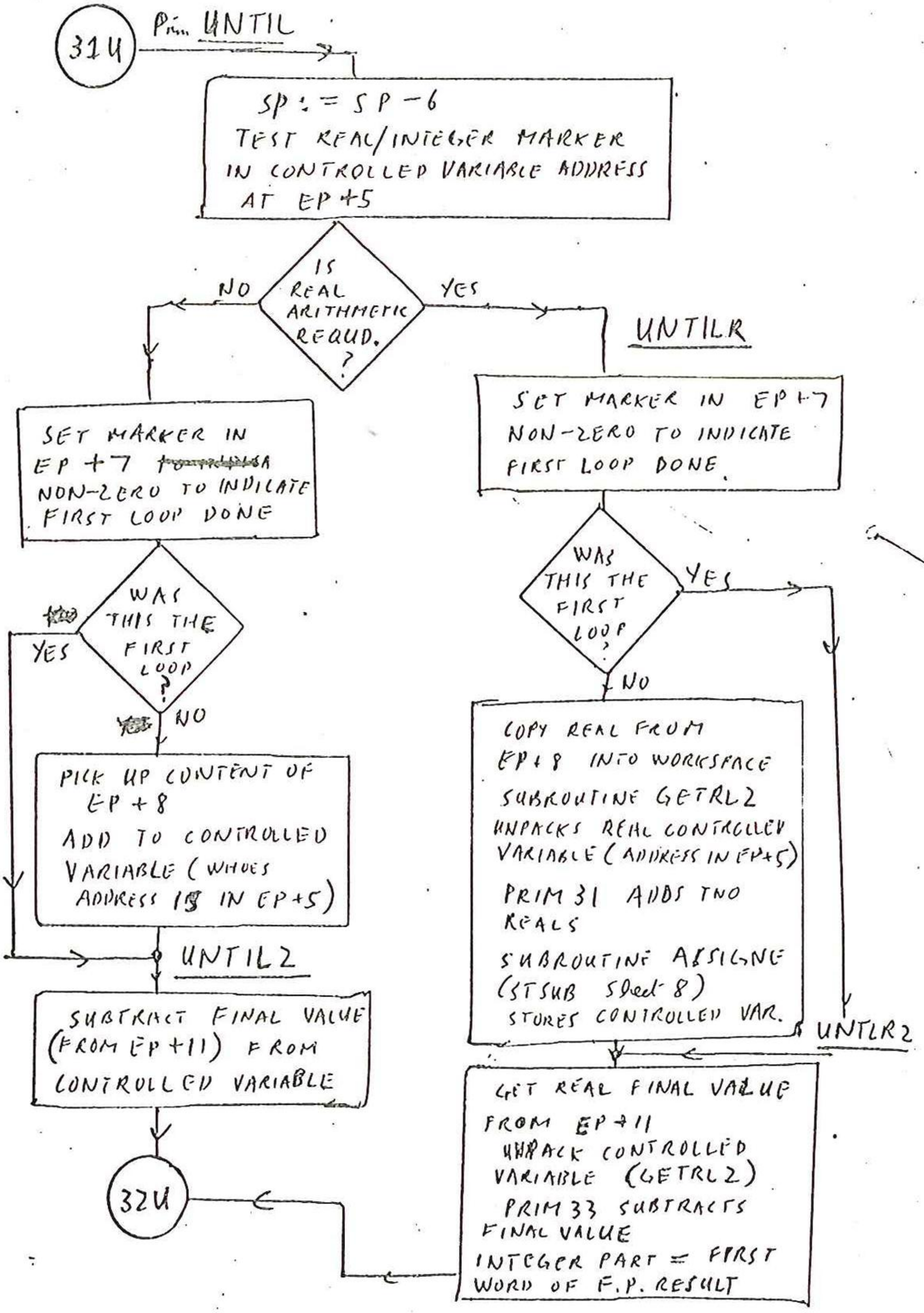
ASSIGNE SUBROUTINE
(STSUBRT +1 Sheet 8)

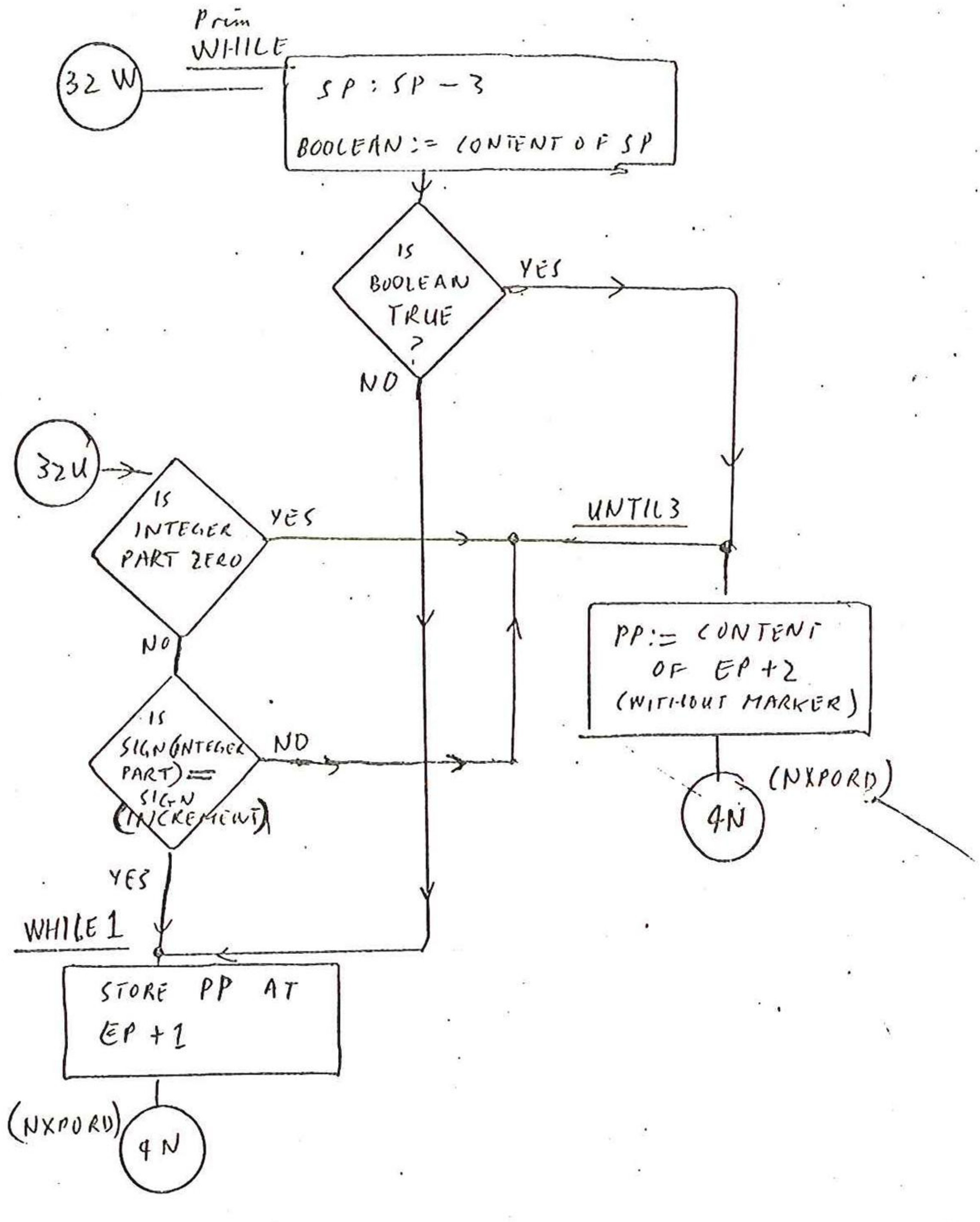
(NXPORD) 4N

30N — Prim NEGR

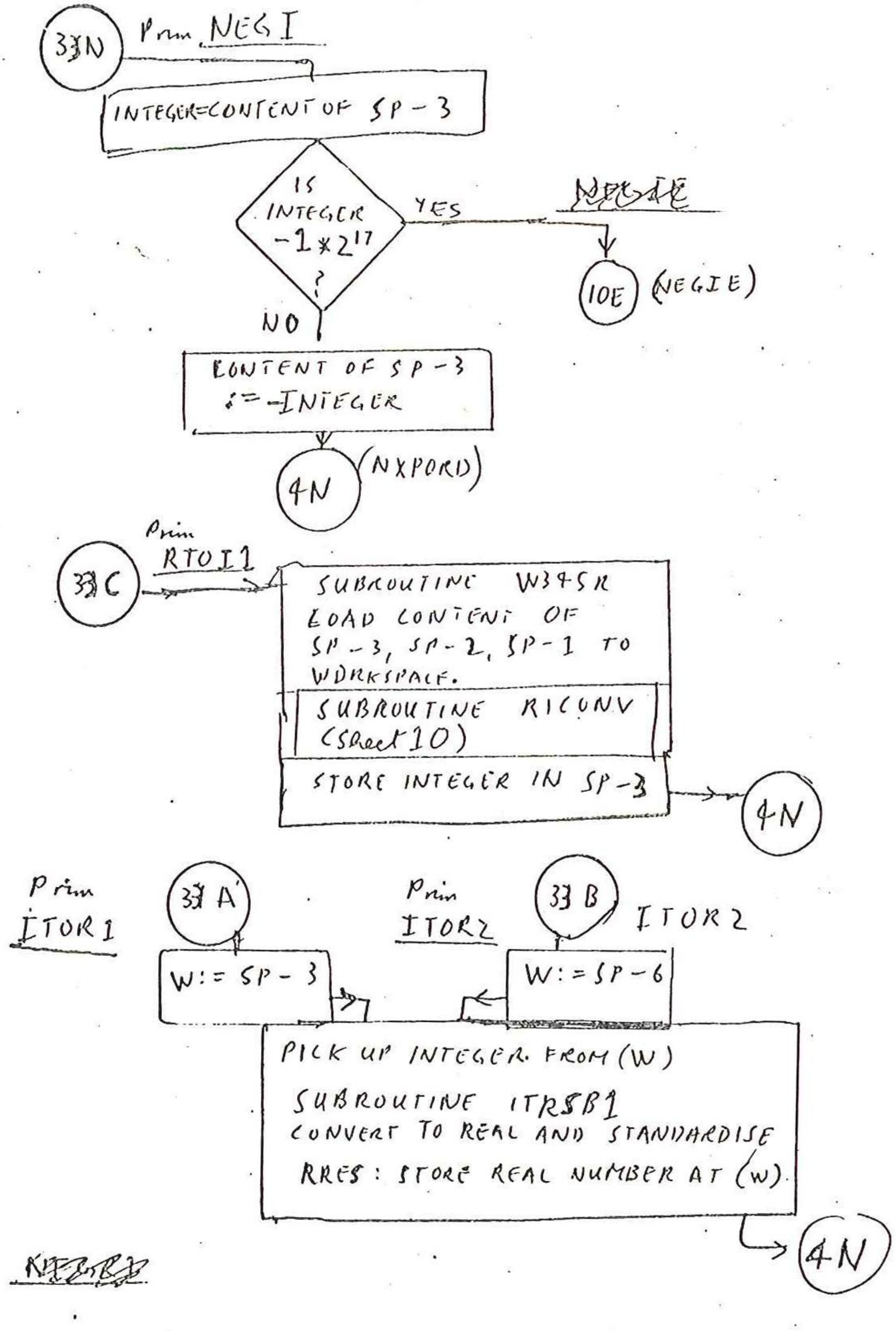
ADDRESS OF REAL := SP−3
SUBROUTINE NEGR1
(Sheet 11)

4N

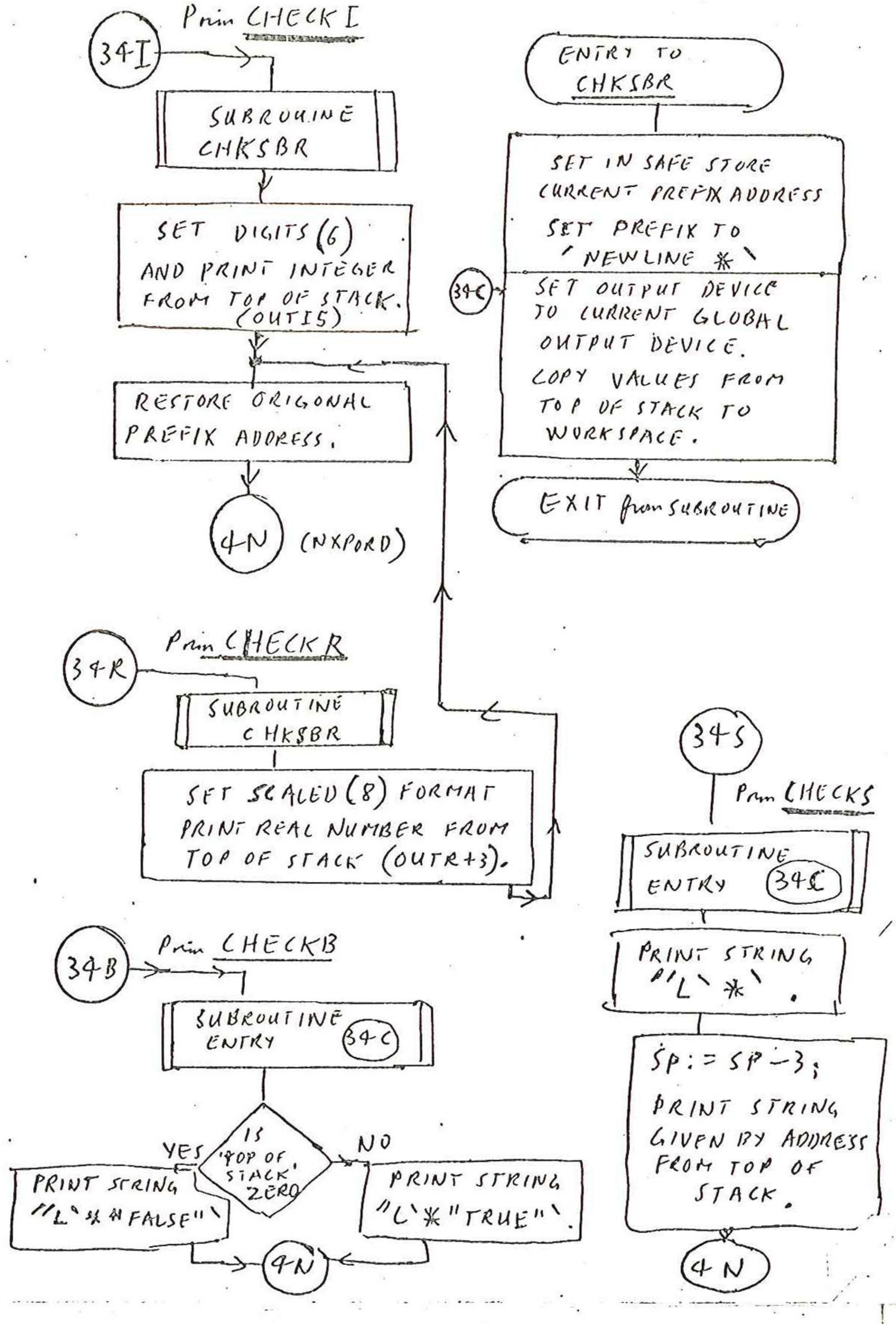

31U
Prim UNTIL
SP := SP - 6
TEST REAL/INTEGER MARKER IN CONTROLLED VARIABLE ADDRESS AT EP+5
IS REAL ARITHMETIC REQUD. ?
NO
YES
UNTILR
SET MARKER IN EP+7 ~~[illegible]~~ NON-ZERO TO INDICATE FIRST LOOP DONE
SET MARKER IN EP+7 NON-ZERO TO INDICATE FIRST LOOP DONE.
WAS THIS THE FIRST LOOP ?
YES
NO
WAS THIS THE FIRST LOOP ?
YES
NO
PICK UP CONTENT OF EP+8
ADD TO CONTROLLED VARIABLE (WHOES ADDRESS IS IN EP+5)
COPY REAL FROM EP+8 INTO WORKSPACE
SUBROUTINE GETRL2 UNPACKS REAL CONTROLLED VARIABLE (ADDRESS IN EP+5)
PRIM 31 ADDS TWO REALS
SUBROUTINE ASSIGNE (STSUB Sheet 8) STORES CONTROLLED VAR.
UNTIL2
UNTLR2
SUBTRACT FINAL VALUE (FROM EP+11) FROM CONTROLLED VARIABLE
GET REAL FINAL VALUE FROM EP+11
UNPACK CONTROLLED VARIABLE (GETRL2)
PRIM 33 SUBTRACTS FINAL VALUE
INTEGER PART = FIRST WORD OF F.P. RESULT
32U


Prim
WHILE
32 W
SP : SP − 3
BOOLEAN := CONTENT OF SP
IS BOOLEAN TRUE ?
YES
NO
32U
IS INTEGER PART ZERO
YES
UNTIL3
NO
PP := CONTENT OF EP + 2 (WITHOUT MARKER)
4N
(NXPORD)
IS SIGN(INTEGER PART) = SIGN (INCREMENT)
NO
YES
WHILE 1
STORE PP AT EP + 1
(NXPORD)
4 N

Sheet 33


33N
Prim. NEGI
INTEGER=CONTENT OF SP-3
IS INTEGER $-1*2^{17}$?
YES
NO
10E
(NEGIE)
CONTENT OF SP-3 := -INTEGER
4N
(NXPORD)
33C
Prim RTOI1
SUBROUTINE W34SR
LOAD CONTENT OF SP-3, SP-2, SP-1 TO WORKSPACE.
SUBROUTINE RICONV (Sheet 10)
STORE INTEGER IN SP-3
4N
Prim ITOR1
33A
Prim ITOR2
33B
ITOR2
W:=SP-3
W:=SP-6
PICK UP INTEGER FROM (W)
SUBROUTINE ITRSB1
CONVERT TO REAL AND STANDARDISE
RRES: STORE REAL NUMBER AT (W).
4N

Sheet 34


Prim CHECK I
34I
SUBROUTINE CHKSBR
SET DIGITS (6) AND PRINT INTEGER FROM TOP OF STACK. (OUTI5)
RESTORE ORIGONAL PREFIX ADDRESS.
4N (NXPORD)
ENTRY TO CHKSBR
SET IN SAFE STORE CURRENT PREFIX ADDRESS
SET PREFIX TO 'NEWLINE * '
34C
SET OUTPUT DEVICE TO CURRENT GLOBAL OUTPUT DEVICE.
COPY VALUES FROM TOP OF STACK TO WORKSPACE.
EXIT from SUBROUTINE
Prim CHECK R
34R
SUBROUTINE CHKSBR
SET SCALED (8) FORMAT PRINT REAL NUMBER FROM TOP OF STACK (OUTR+3).
34S
Prim CHECKS
SUBROUTINE ENTRY 34C
PRINT STRING "L` * ` .
SP: = SP - 3;
PRINT STRING GIVEN BY ADDRESS FROM TOP OF STACK.
4N
Prim CHECKB
34B
SUBROUTINE ENTRY 34C
IS 'TOP OF STACK' ZERO
YES
NO
PRINT STRING "L` * "FALSE" `
PRINT STRING "L` * "TRUE" `.
4N

(35F) Prim FINISH

OUTPUT 20 BLANKS ON PUNCH
OUTPUT DEVICE (ODEV) :=
CURRENT GLOBAL OUTPUT DEVICE (GLODEV)
PRINT STRING "L2' FINISH "L2"

SUBROUTINE FINSUB

OUTPUT 150 BLANKS TO PUNCH

(35S) STOP

SET CONTINUE ADDRESS TO (STOP)

(7W) (SWAIT)

(35N) Prim NEGR

W:=SP-3 (ADDRESS OF REAL)

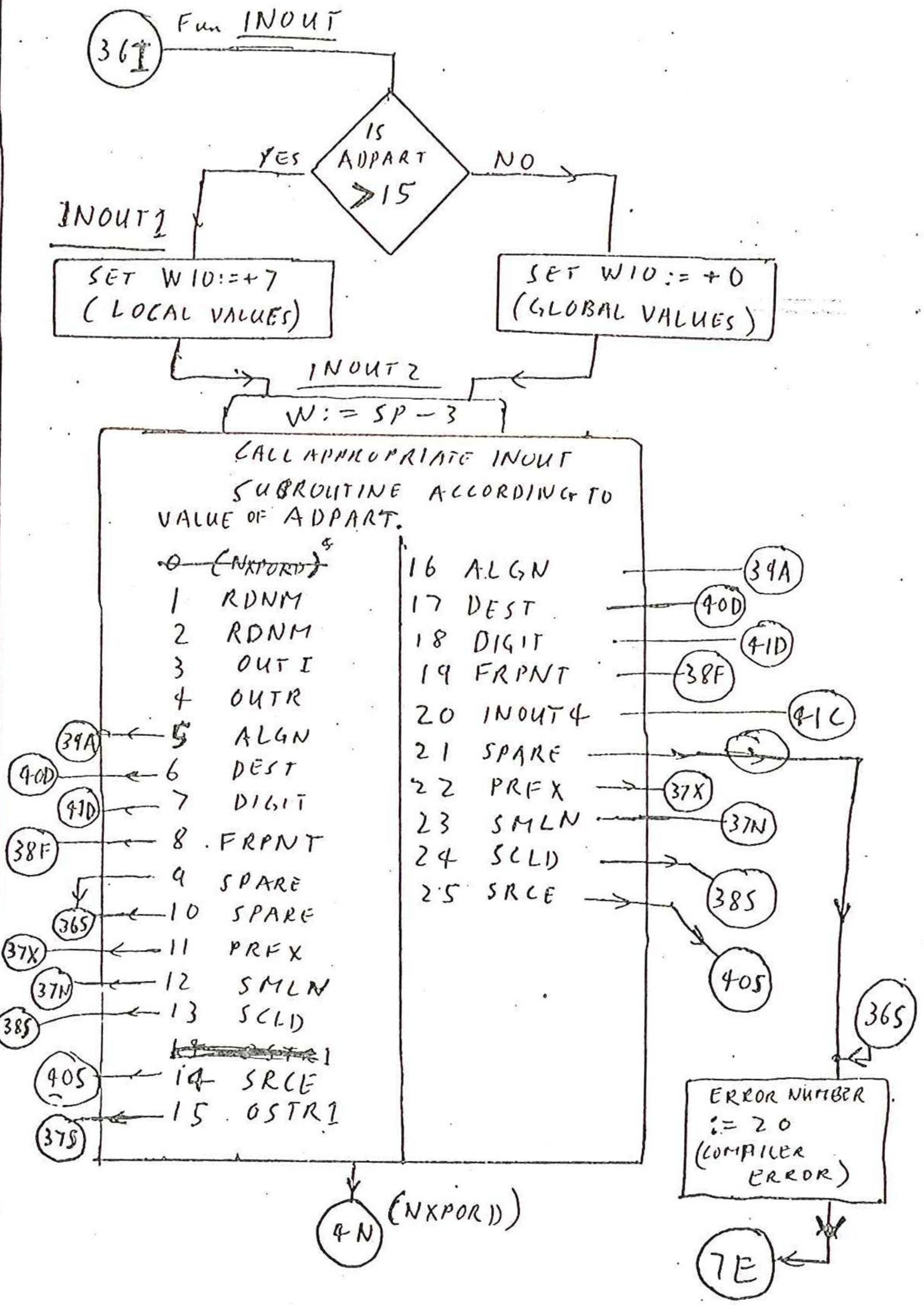
SUBROUTINE NEGR1 (Sheet 11)

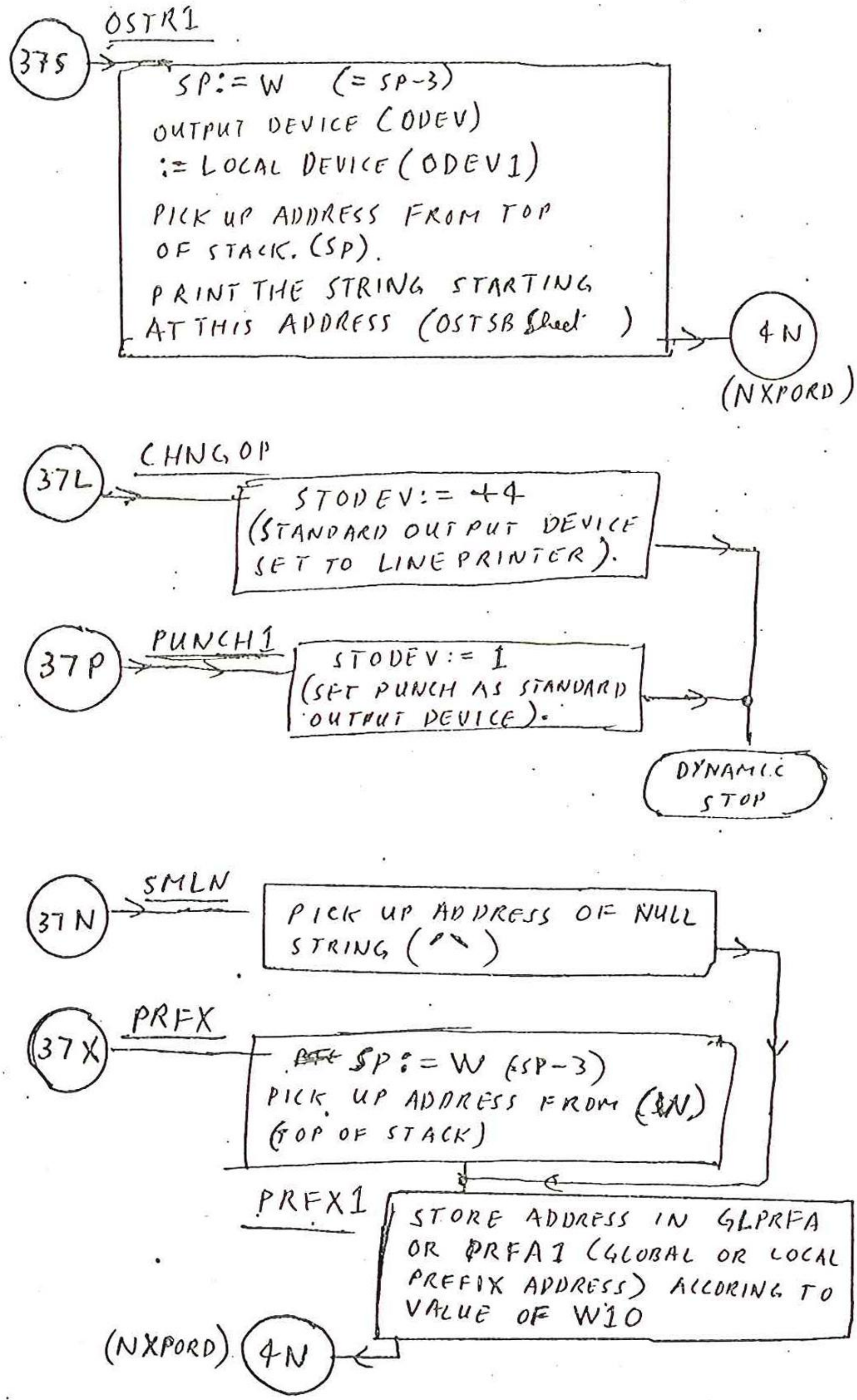
(4N) (NXPORD)

(35W) Prim WAIT

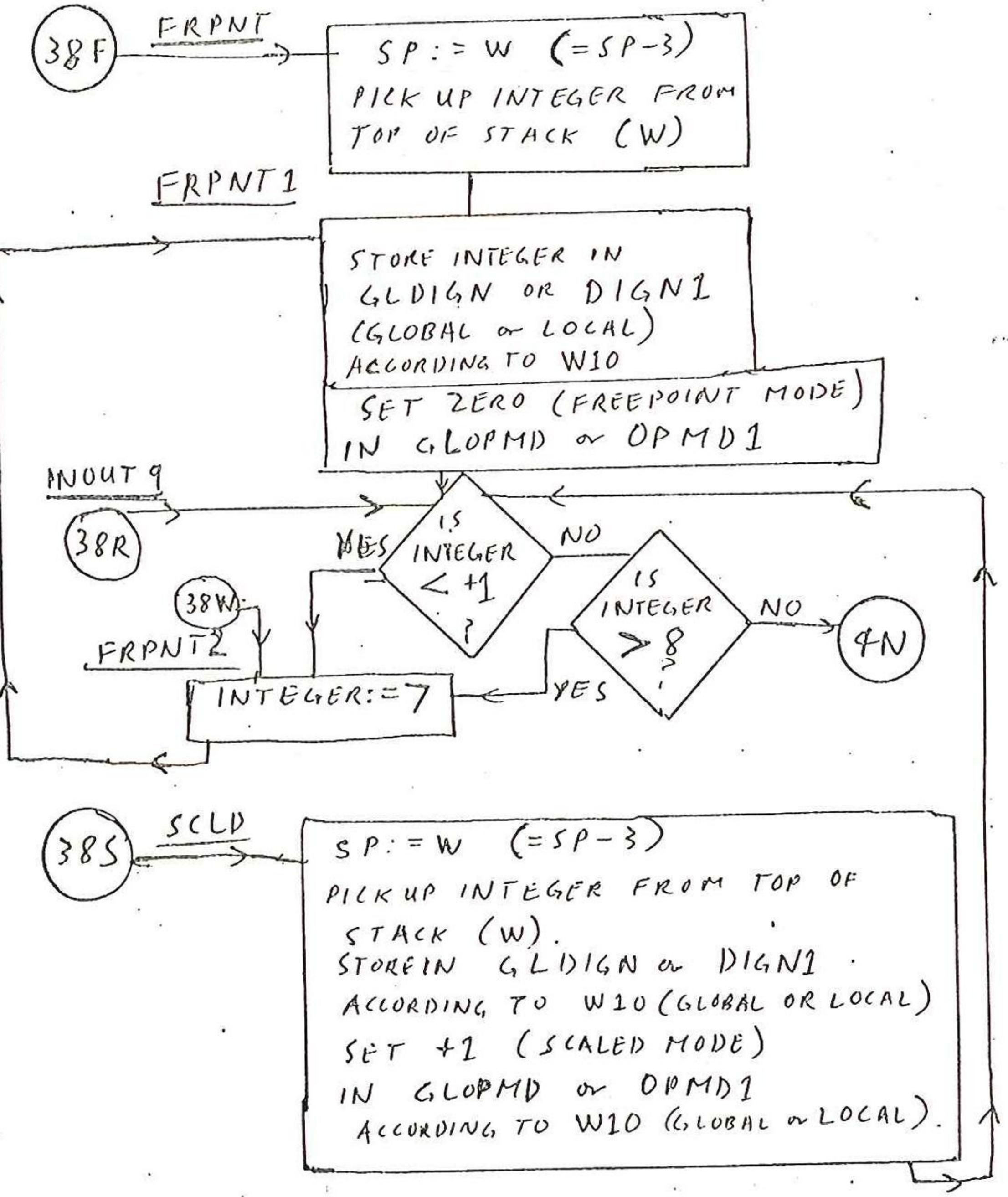
SET CONTINUE ADDRESS IN PAUSRT TO GOTO NXPORD (Sheet 4)

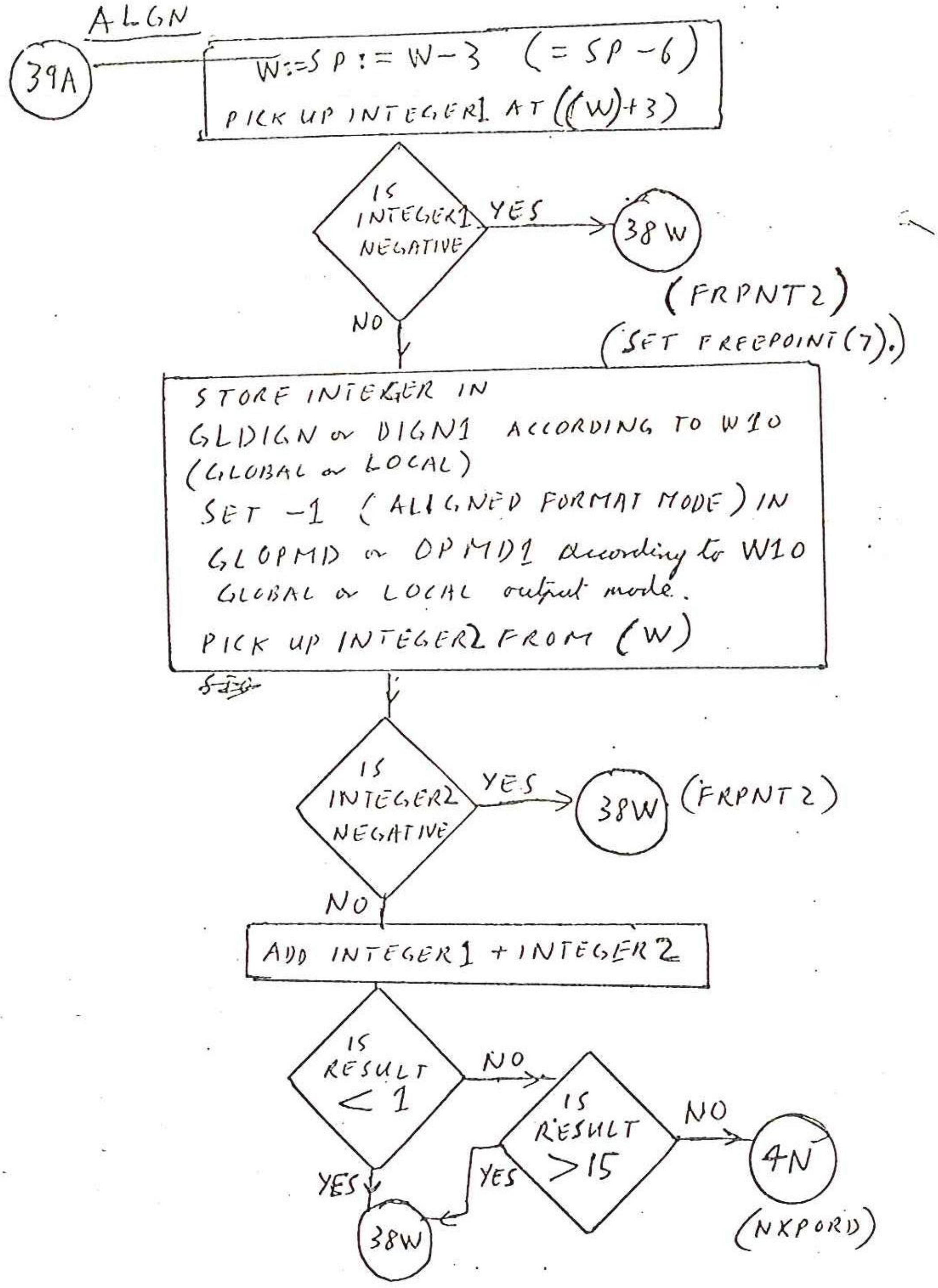
(SWAIT) (7W)


36I
Fun INOUT
IS ADPART >15
YES
NO
INOUT1
SET W10:=+7 (LOCAL VALUES)
SET W10:=+0 (GLOBAL VALUES)
INOUT2
W:=SP-3
CALL APPROPRIATE INOUT SUBROUTINE ACCORDING TO VALUE OF ADPART.
~~0 (NXPORD)~~
1 RDNM
2 RDNM
3 OUTI
4 OUTR
5 ALGN
6 DEST
7 DIGIT
8 FRPNT
9 SPARE
10 SPARE
11 PRFX
12 SMLN
13 SCLD
14 SRCE
15 OSTR1
16 ALGN
17 DEST
18 DIGIT
19 FRPNT
20 INOUT4
21 SPARE
22 PRFX
23 SMLN
24 SCLD
25 SRCE
39A
40D
41D
38F
41C
37X
37N
38S
40S
36S
37S
ERROR NUMBER := 20 (COMPILER ERROR)
4N
(NXPORD)
7E

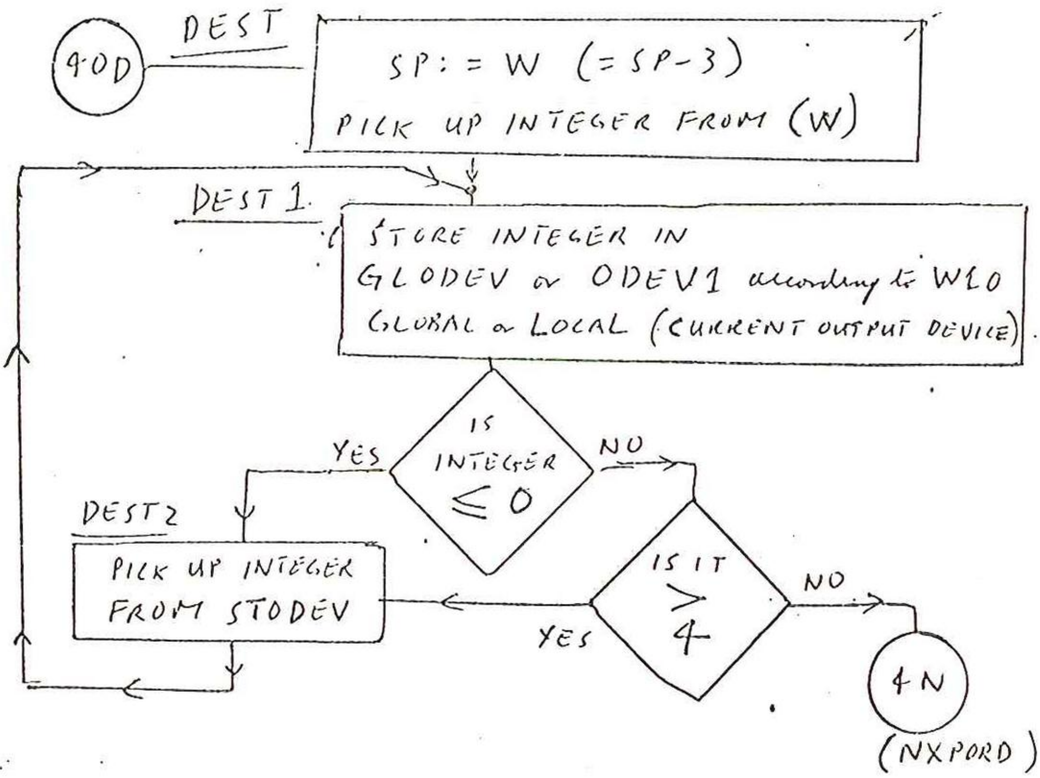

37S
OSTR1
SP:= W (= SP-3)
OUTPUT DEVICE (ODEV)
:= LOCAL DEVICE (ODEV1)
PICK UP ADDRESS FROM TOP
OF STACK. (SP).
PRINT THE STRING STARTING
AT THIS ADDRESS (OSTSB Sheet)
4N
(NXPORD)
37L
CHNGOP
STODEV:= +4
(STANDARD OUTPUT DEVICE
SET TO LINE PRINTER).
37P
PUNCH1
STODEV:= 1
(SET PUNCH AS STANDARD
OUTPUT DEVICE).
DYNAMIC
STOP
37N
SMLN
PICK UP ADDRESS OF NULL
STRING (^)
37X
PRFX
~~PRF~~ SP:= W (SP-3)
PICK UP ADDRESS FROM (SN)
(TOP OF STACK)
PRFX1
STORE ADDRESS IN GLPRFA
OR PRFA1 (GLOBAL OR LOCAL
PREFIX ADDRESS) ACCORING TO
VALUE OF W10
(NXPORD)
4N

Sheet 38

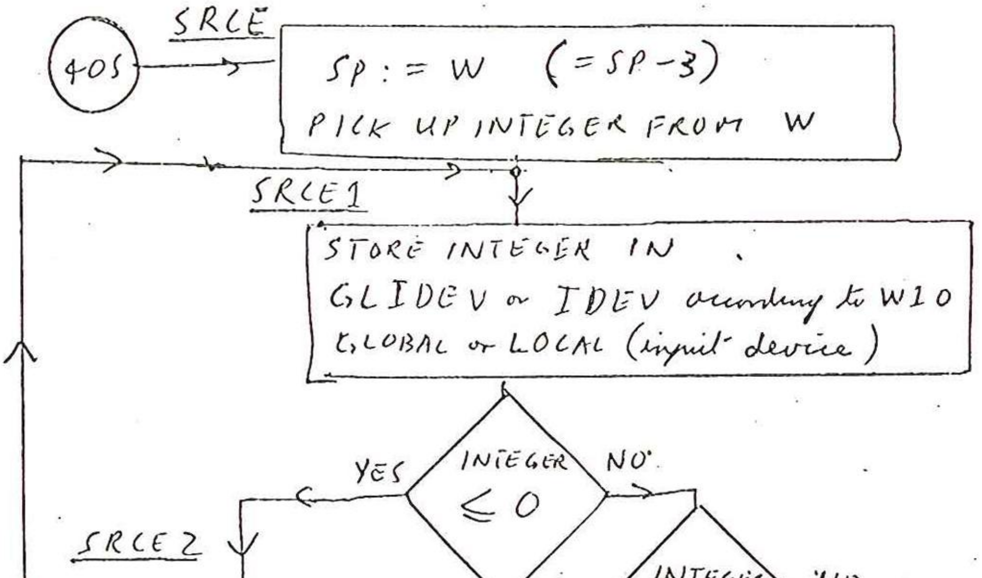

38F
FRPNT
SP:= W (=SP-3)
PICK UP INTEGER FROM TOP OF STACK (W)
FRPNT1
STORE INTEGER IN GLDIGN OR DIGN1 (GLOBAL or LOCAL) ACCORDING TO W10
SET ZERO (FREEPOINT MODE) IN GLOPMD or OPMD1
INOUT 9
38R
IS INTEGER < +1 ?
YES
NO
IS INTEGER > 8 ?
NO
4N
YES
38W
FRPNT2
INTEGER:= 7
38S
SCLD
SP:= W (=SP-3)
PICK UP INTEGER FROM TOP OF STACK (W).
STORE IN GLDIGN or DIGN1
ACCORDING TO W10 (GLOBAL OR LOCAL)
SET +1 (SCALED MODE)
IN GLOPMD or OPMD1
ACCORDING TO W10 (GLOBAL or LOCAL).


ALGN
39A
W:=SP:= W−3 (= SP−6)
PICK UP INTEGER1 AT ((W)+3)
IS INTEGER1 NEGATIVE
YES
38W
(FRPNT2)
(SET FREEPOINT(7).)
NO
STORE INTEGER IN
GLDIGN or DIGN1 ACCORDING TO W10
(GLOBAL or LOCAL)
SET −1 (ALIGNED FORMAT MODE) IN
GLOPMD or OPMD1 according to W10
GLOBAL or LOCAL output mode.
PICK UP INTEGER2 FROM (W)
IS INTEGER2 NEGATIVE
YES
38W
(FRPNT2)
NO
ADD INTEGER1 + INTEGER2
IS RESULT < 1
NO
IS RESULT > 15
NO
4N
(NXPORD)
YES
YES
38W

Sheet 40

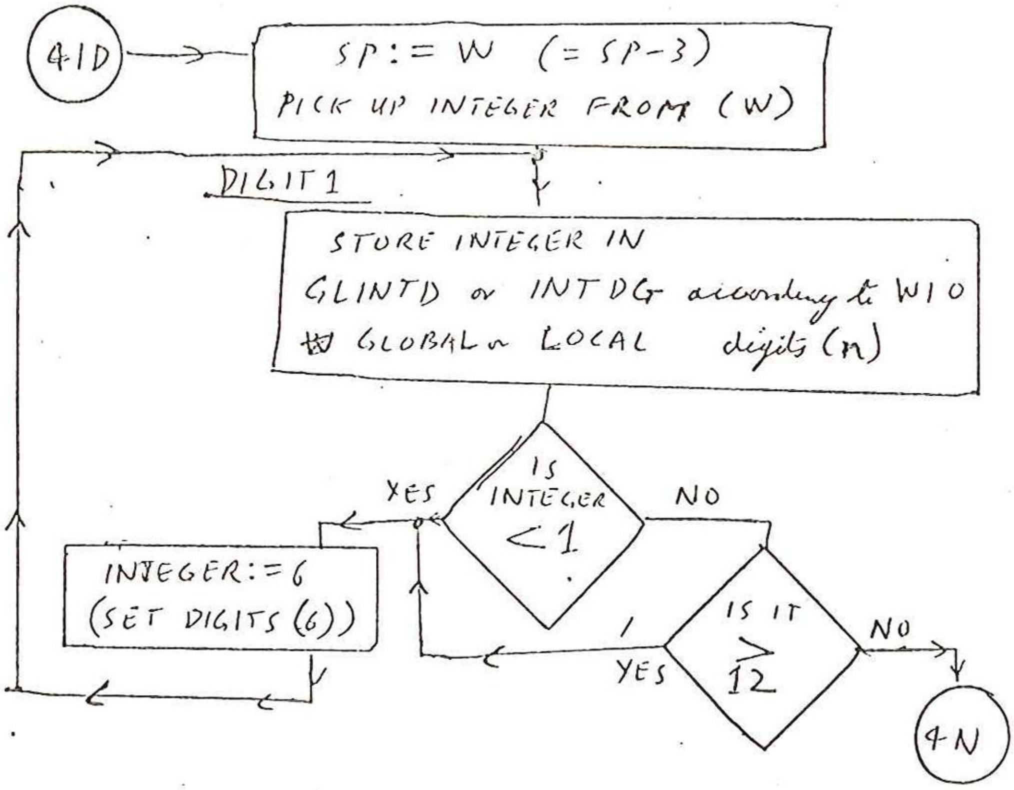


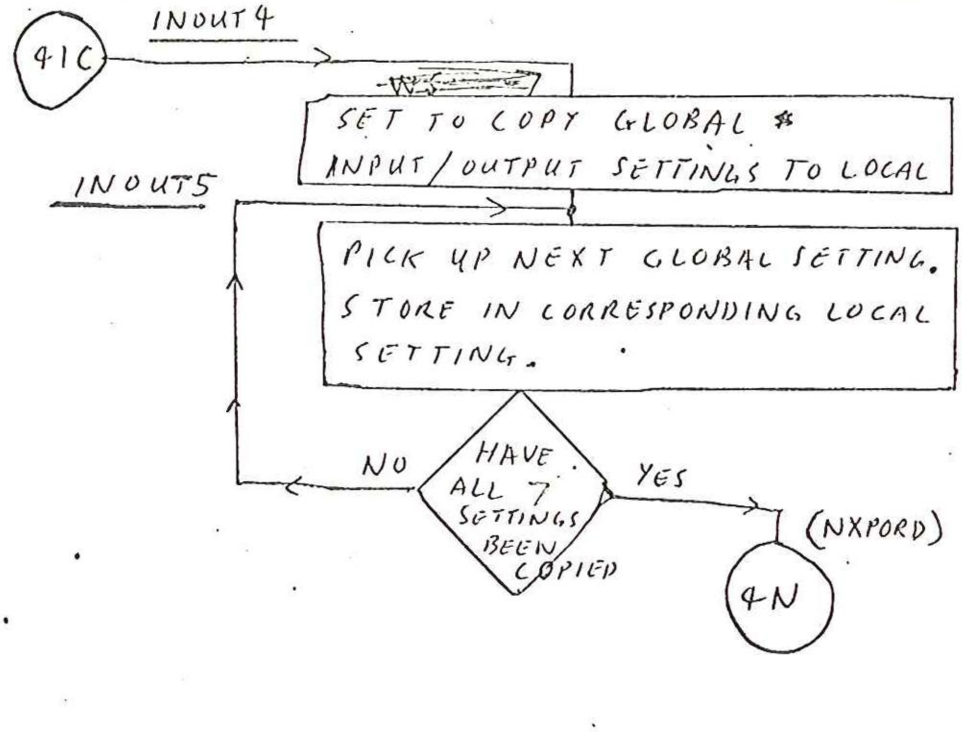
40D
DEST
SP: = W (= SP-3)
PICK UP INTEGER FROM (W)
DEST 1.
STORE INTEGER IN
GLODEV or ODEV1 according to W10
GLOBAL or LOCAL (CURRENT OUTPUT DEVICE)
IS INTEGER ≤ 0
YES
NO
DEST2
PICK UP INTEGER FROM STODEV
IS IT > 4
YES
NO
4N
(NXPORD)



40S
SRCE
SP: = W (= SP-3)
PICK UP INTEGER FROM W
SRCE1
STORE INTEGER IN
GLIDEV or IDEV according to W10
GLOBAL or LOCAL (input device)
INTEGER ≤ 0
YES
NO
SRCE2
INTEGER > 3
YES
NO
INTEGER: = 1
SET READER(1)
4N

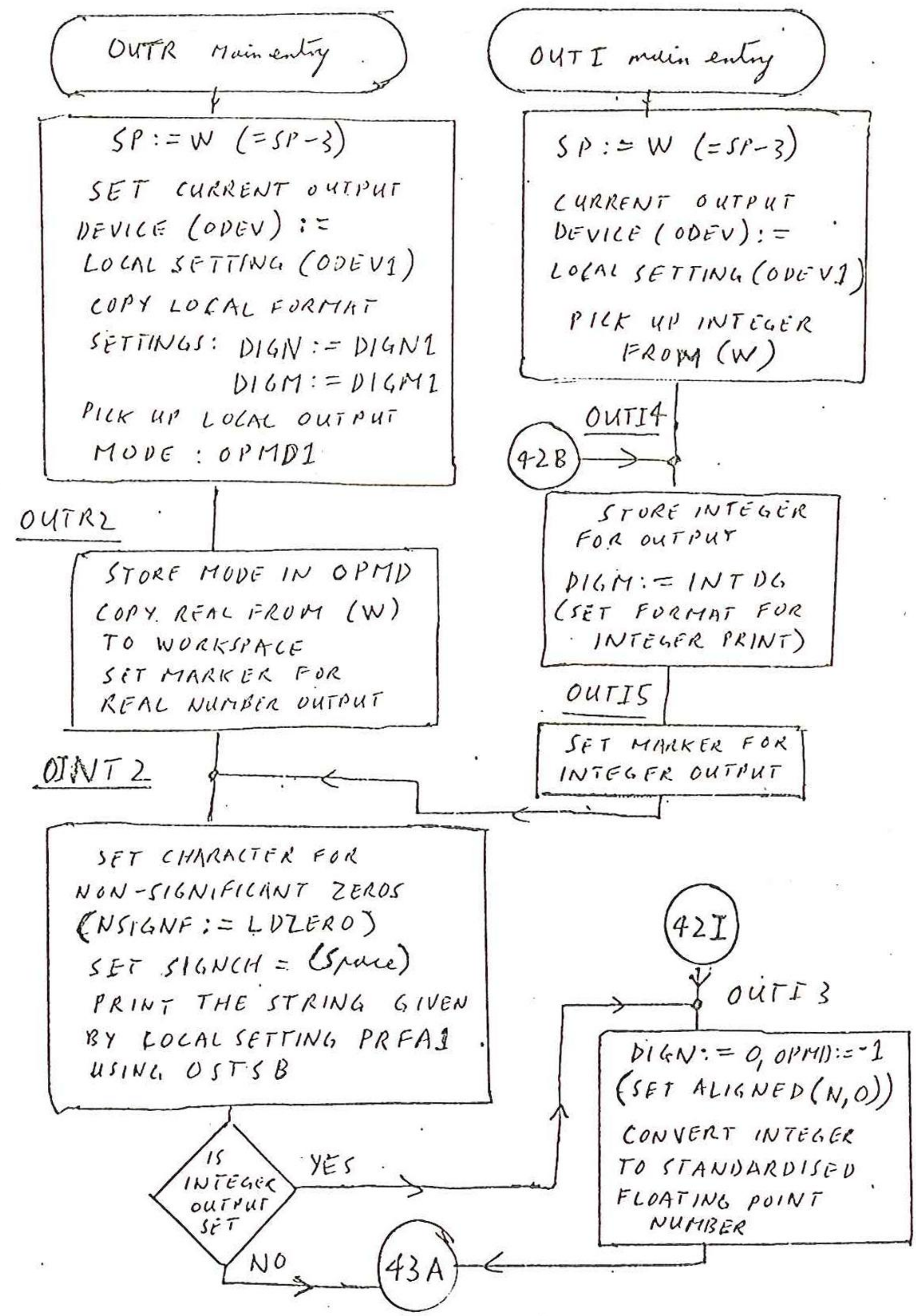
DIGIT


41D
SP:= W (= SP-3)
PICK UP INTEGER FROM (W)
DIGIT1
STORE INTEGER IN
GLINTD or INTDG according to W10
GLOBAL or LOCAL digits (n)
IS INTEGER < 1
YES
NO
INTEGER:= 6
(SET DIGITS (6))
IS IT > 12
YES
NO
4N

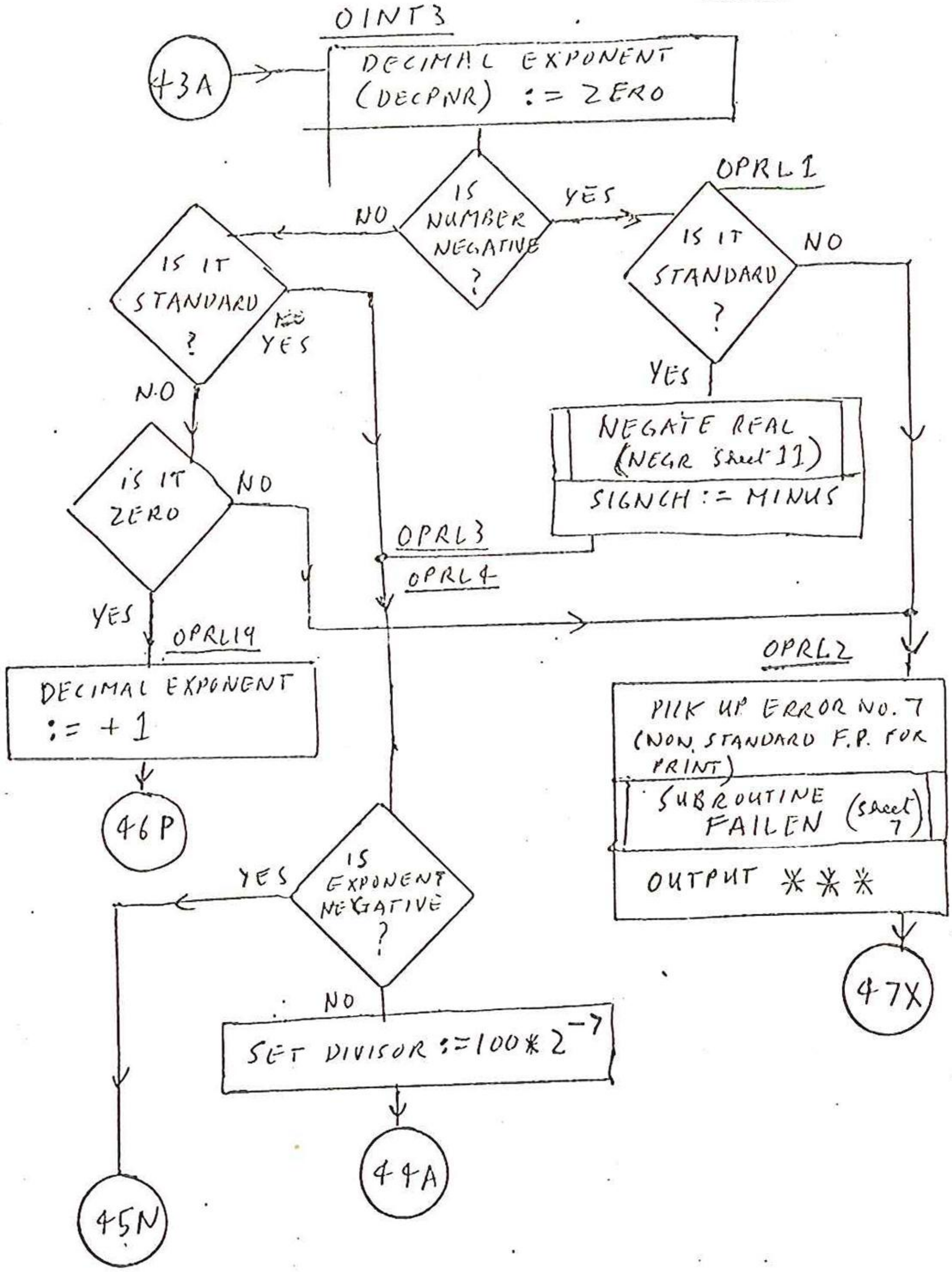

41C
INOUT4
SET TO COPY GLOBAL
INPUT/OUTPUT SETTINGS TO LOCAL
INOUT5
PICK UP NEXT GLOBAL SETTING.
STORE IN CORRESPONDING LOCAL
SETTING.
HAVE ALL 7 SETTINGS BEEN COPIED
NO
YES
(NXPORD)
4N

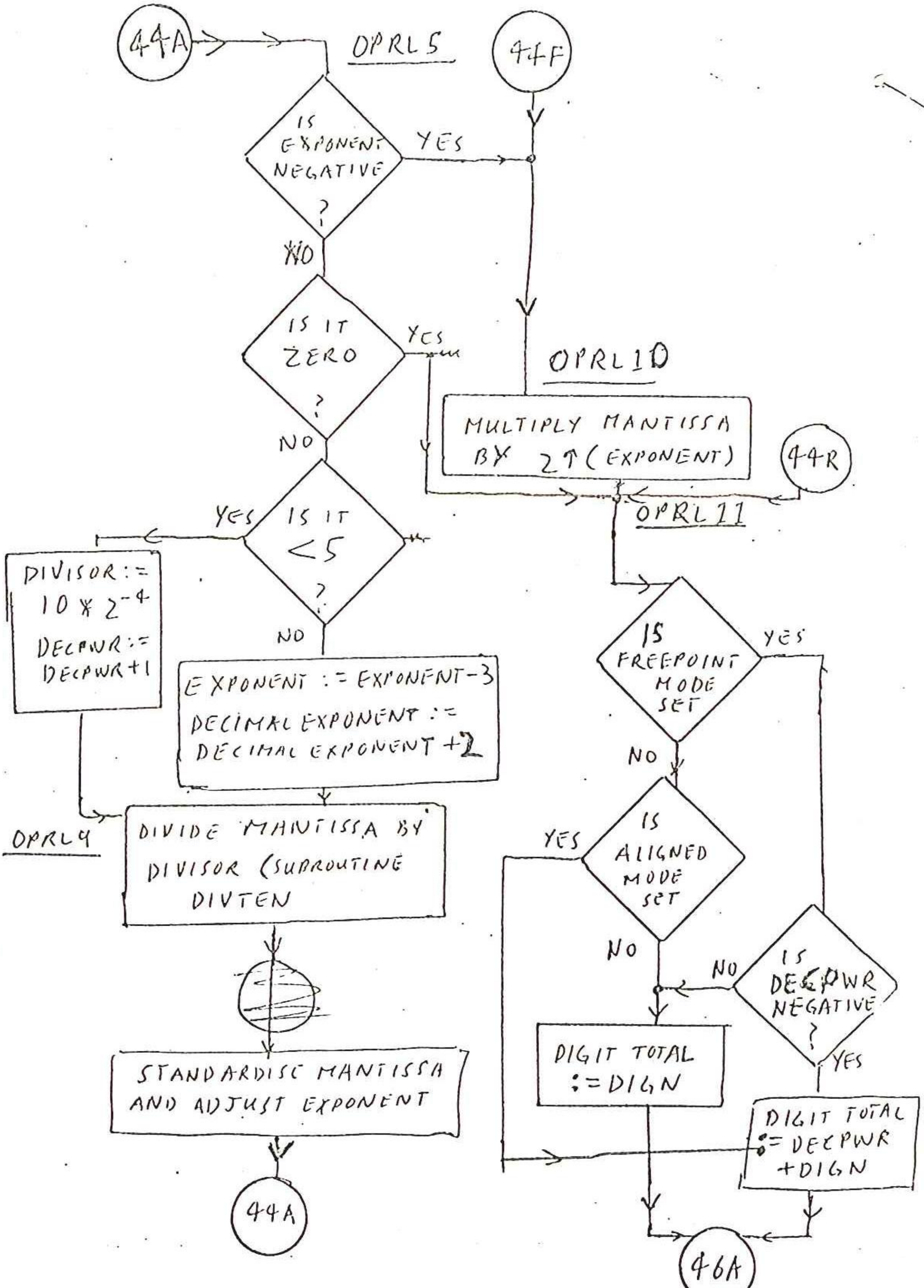
Sheet 42

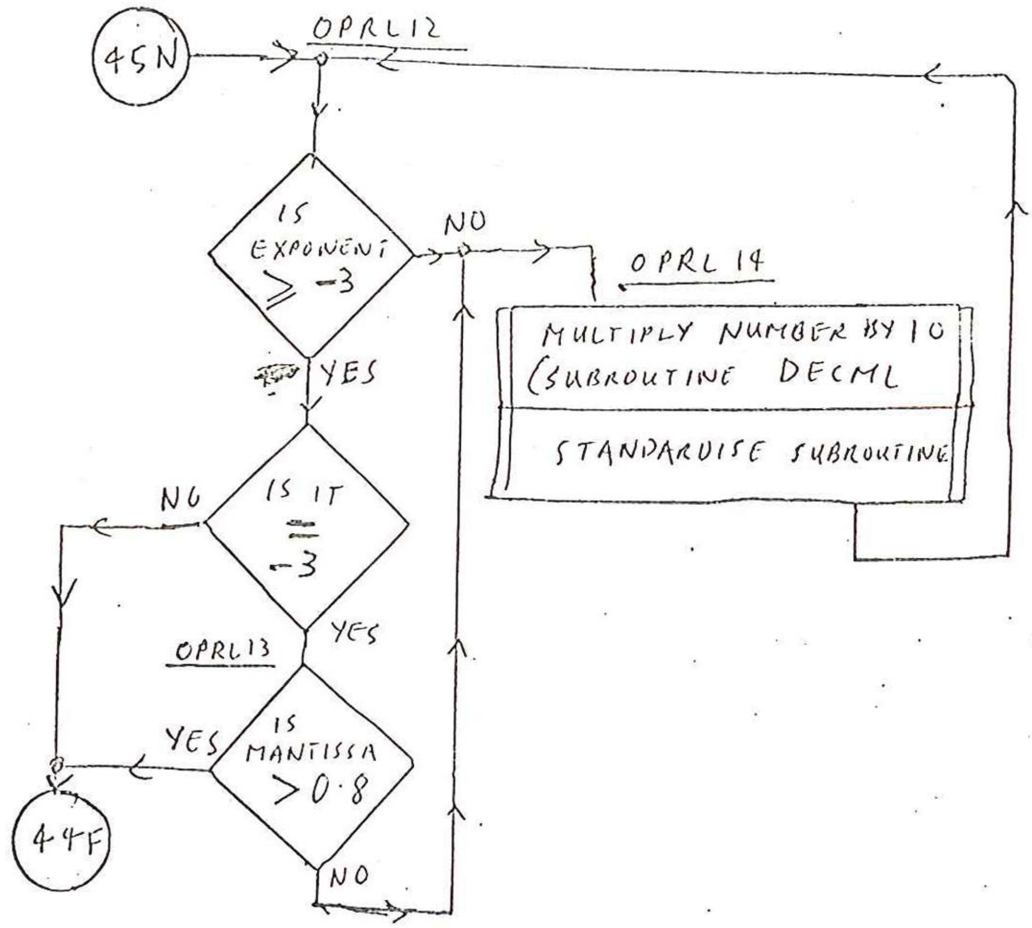
OUTPUT SUBROUTINE



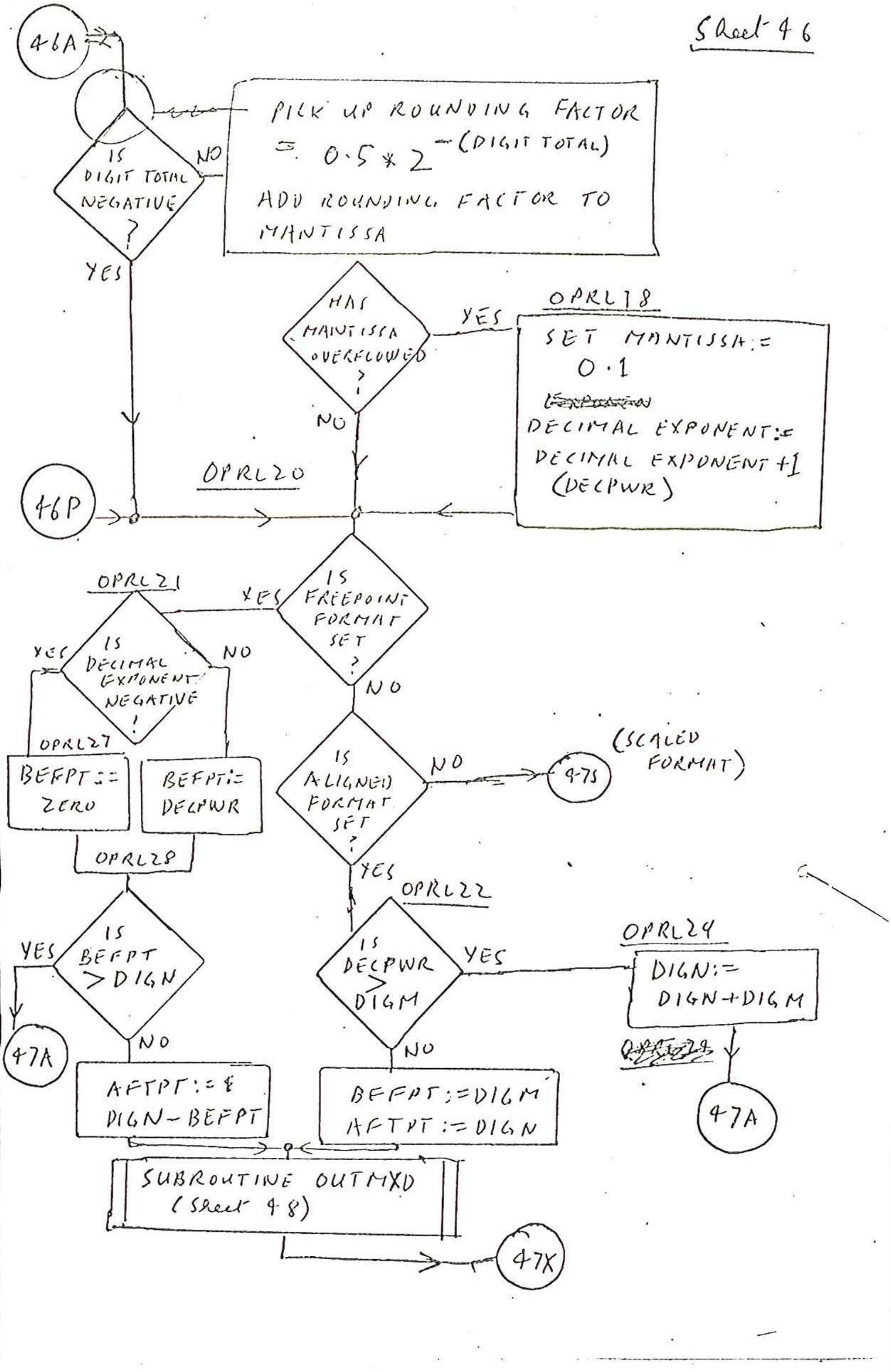
Sheet 43


OINT3
43A
DECIMAL EXPONENT (DECPNR) := ZERO
IS NUMBER NEGATIVE ?
NO
YES
OPRL1
IS IT STANDARD ?
NO
YES
NEGATE REAL (NEGR Sheet 11)
SIGNCH := MINUS
IS IT STANDARD ?
YES
N.O
IS IT ZERO
NO
YES
OPRL3
OPRL4
OPRL19
DECIMAL EXPONENT := +1
46P
OPRL2
PICK UP ERROR NO. 7 (NON STANDARD F.P. FOR PRINT)
SUBROUTINE FAILEN (Sheet 7)
OUTPUT ***
47X
IS EXPONENT NEGATIVE ?
YES
NO
SET DIVISOR := 100 * 2^-7
44A
45N

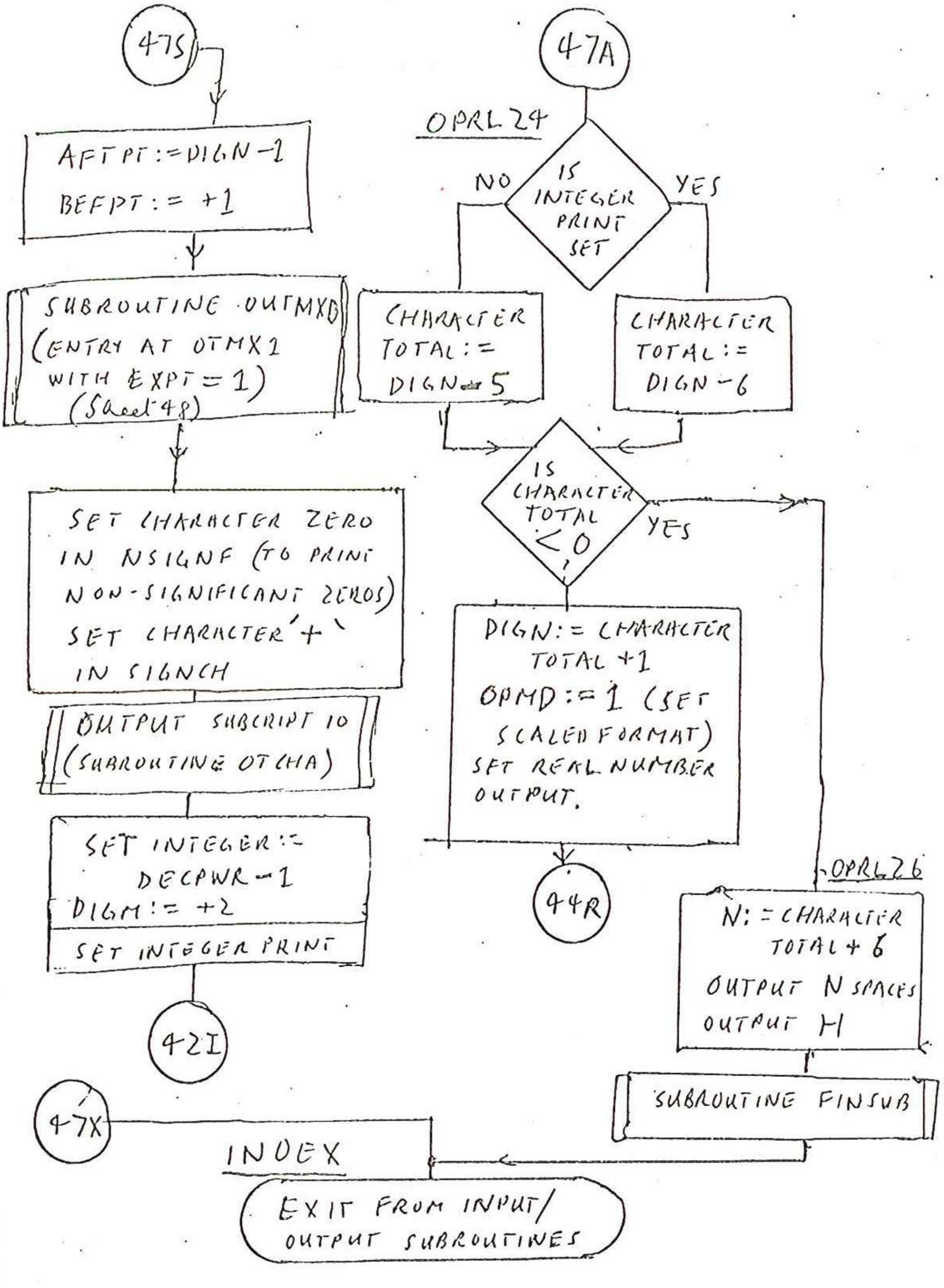

44A
OPRL5
44F
IS EXPONENT NEGATIVE ?
YES
NO
IS IT ZERO ?
YES
NO
OPRL10
MULTIPLY MANTISSA BY 2↑(EXPONENT)
44R
OPRL11
IS IT <5 ?
YES
NO
DIVISOR:= 10 * 2^-4
DECPWR:= DECPWR+1
EXPONENT := EXPONENT-3
DECIMAL EXPONENT := DECIMAL EXPONENT +2
OPRL4
DIVIDE MANTISSA BY DIVISOR (SUBROUTINE DIVTEN
STANDARDISE MANTISSA AND ADJUST EXPONENT
44A
IS FREEPOINT MODE SET
YES
NO
IS ALIGNED MODE SET
YES
NO
IS DECPWR NEGATIVE ?
NO
YES
DIGIT TOTAL := DIGN
DIGIT TOTAL := DECPWR +DIGN
46A


45N
OPRL12
IS EXPONENT ≥ -3
NO
YES
OPRL 14
MULTIPLY NUMBER BY 10 (SUBROUTINE DECML
STANDARDISE SUBROUTINE
IS IT = -3
NO
YES
OPRL13
IS MANTISSA > 0·8
YES
NO
44F

Sheet 46

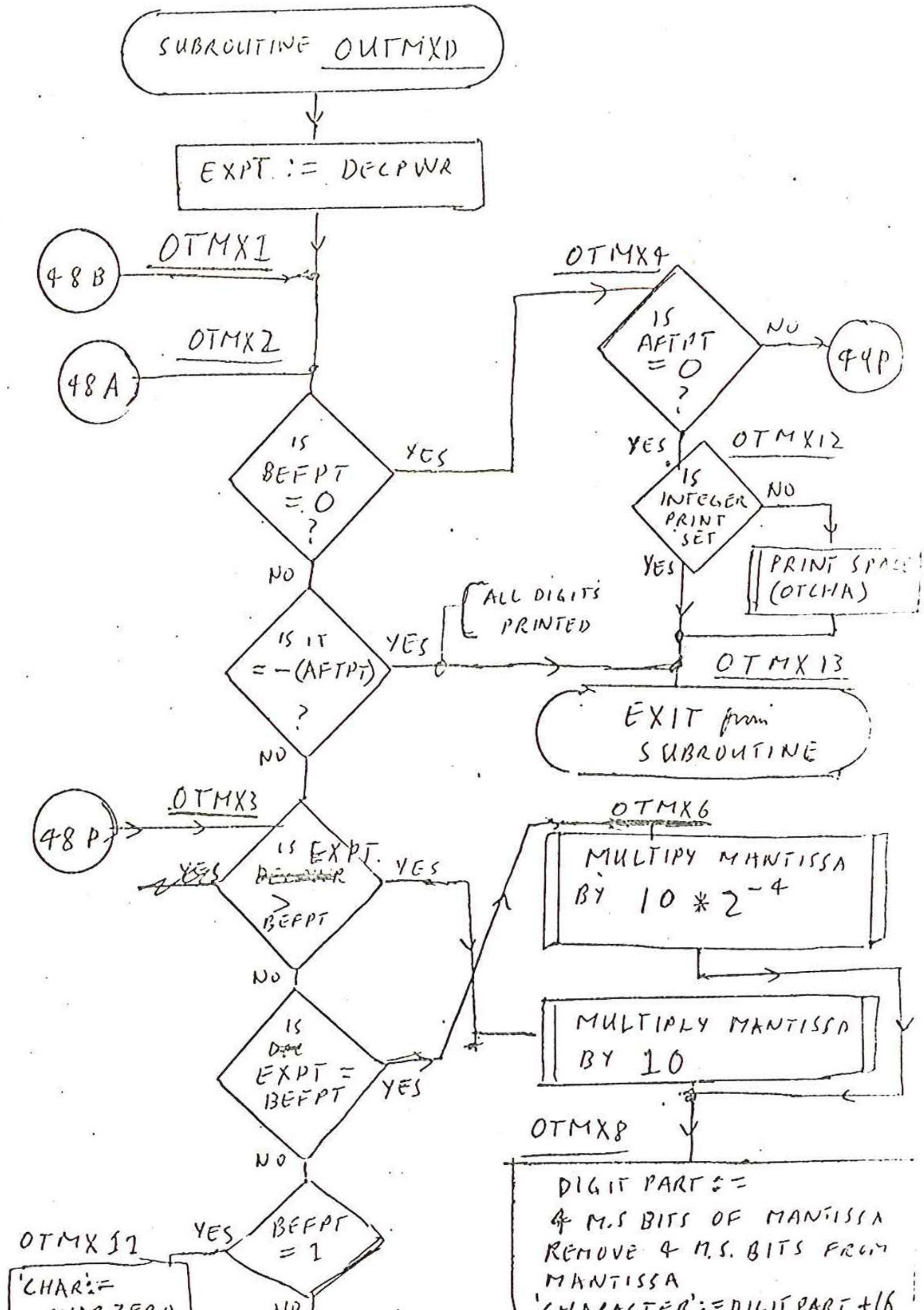

46A
IS DIGIT TOTAL NEGATIVE ?
NO
YES
PICK UP ROUNDING FACTOR = 0.5 * 2 −(DIGIT TOTAL)
ADD ROUNDING FACTOR TO MANTISSA
HAS MANTISSA OVERFLOWED ?
YES
NO
OPRL18
SET MANTISSA:= 0.1
~~EXPONENT~~
DECIMAL EXPONENT:= DECIMAL EXPONENT +1 (DECPWR)
46P
OPRL20
IS FREEPOINT FORMAT SET ?
YES
NO
OPRL21
IS DECIMAL EXPONENT NEGATIVE ?
YES
NO
OPRL27
BEFPT:= ZERO
BEFPT:= DECPWR
OPRL28
IS BEFPT > DIGN
YES
NO
47A
AFTPT:= DIGN − BEFPT
IS ALIGNED FORMAT SET ?
NO
47S
(SCALED FORMAT)
YES
OPRL22
IS DECPWR > DIGM
YES
NO
OPRL24
DIGN:= DIGN+DIGM
47A
BEFPT:=DIGM
AFTPT:=DIGN
SUBROUTINE OUTMXD (Sheet 48)
47X

Sheet 47

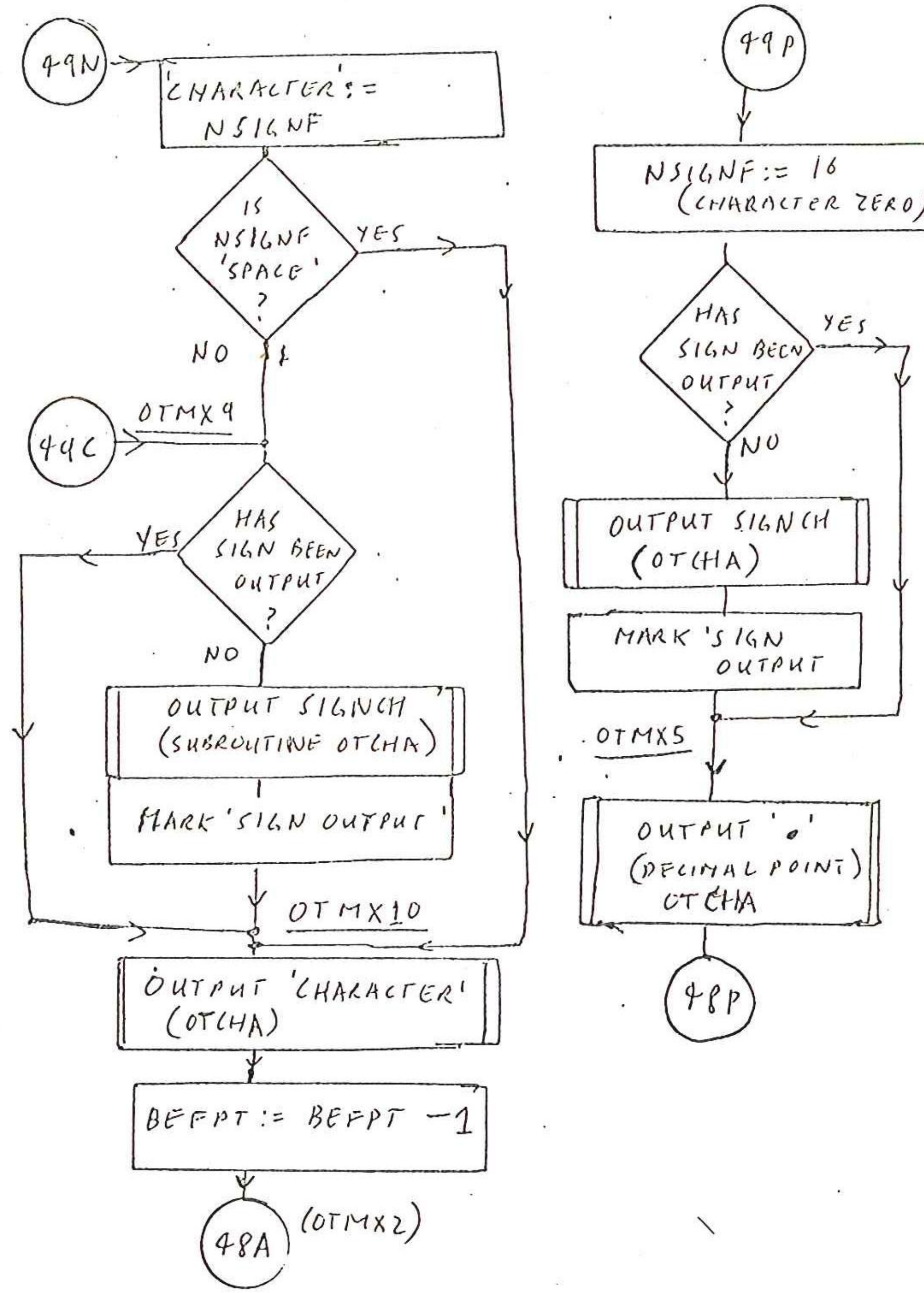


47S
AFTPT:=DIGN−1
BEFPT:= +1
SUBROUTINE OUTMXD (ENTRY AT OTMX1 WITH EXPT = 1) (Sheet 48)
SET CHARACTER ZERO IN NSIGNF (TO PRINT NON-SIGNIFICANT ZEROS) SET CHARACTER '+' IN SIGNCH
OUTPUT SUBCRIPT 10 (SUBROUTINE OTCHA)
SET INTEGER:= DECPWR−1
DIGM:= +2
SET INTEGER PRINT
42I
47A
OPRL24
IS INTEGER PRINT SET
NO
YES
CHARACTER TOTAL:= DIGN−5
CHARACTER TOTAL:= DIGN−6
IS CHARACTER TOTAL <0
YES
DIGN:= CHARACTER TOTAL +1
OPMD:=1 (SET SCALED FORMAT)
SET REAL NUMBER OUTPUT.
44R
OPRL26
N:=CHARACTER TOTAL+6
OUTPUT N SPACES
OUTPUT H
SUBROUTINE FINSUB
47X
INDEX
EXIT FROM INPUT/OUTPUT SUBROUTINES

Sheet 48

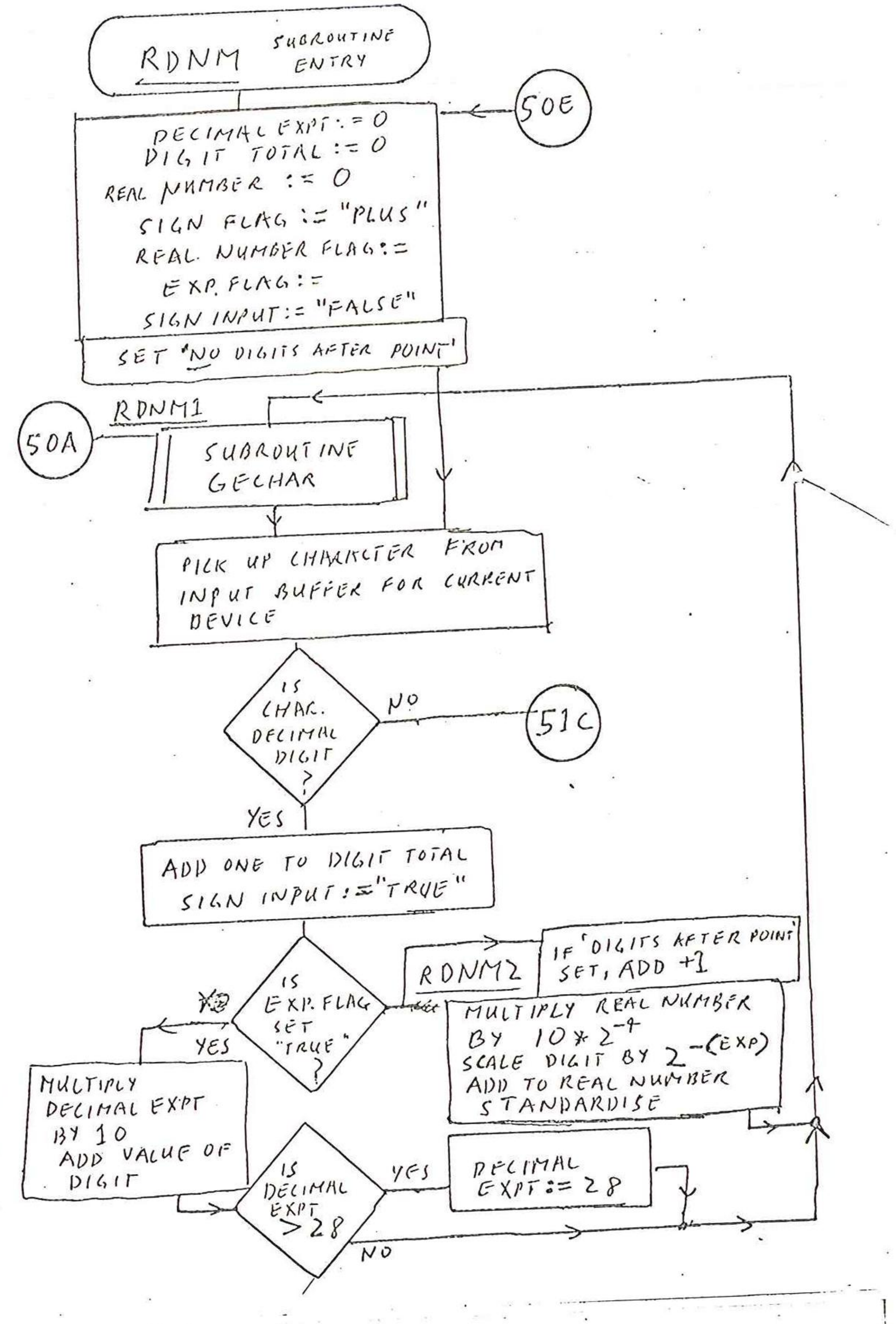

SUBROUTINE OUTMXD
EXPT := DECPWR
OTMX1
48 B
OTMX2
48 A
IS BEFPT = 0 ?
YES
NO
OTMX4
IS AFTPT = 0 ?
NO
44P
YES
OTMX12
IS INTEGER PRINT SET
NO
PRINT SPACE (OTCHA)
YES
ALL DIGITS PRINTED
IS IT = −(AFTPT) ?
YES
NO
OTMX13
EXIT from SUBROUTINE
OTMX3
48 P
IS EXPT > BEFPT
YES
NO
OTMX6
MULTIPY MANTISSA BY 10 * 2^-4
MULTIPLY MANTISSA BY 10
IS EXPT = BEFPT
YES
NO
OTMX8
DIGIT PART :=
4 M.S BITS OF MANTISSA
REMOVE 4 M.S. BITS FROM MANTISSA
'CHARACTER' := DIGIT PART +16
BEFPT = 1
YES
NO
OTMX 11
'CHAR' := CHAR.ZERO
49N
49C

Sheet 49

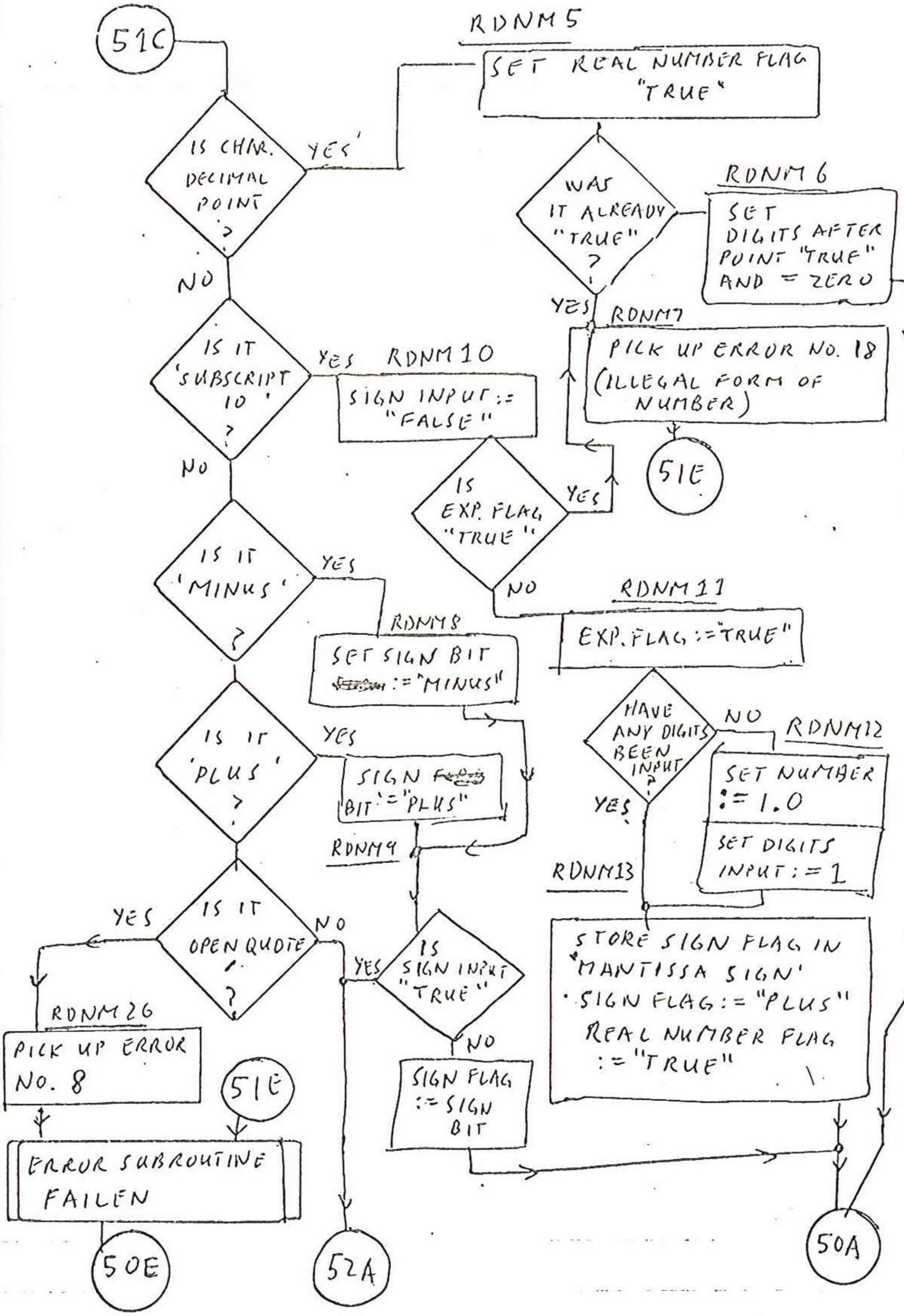


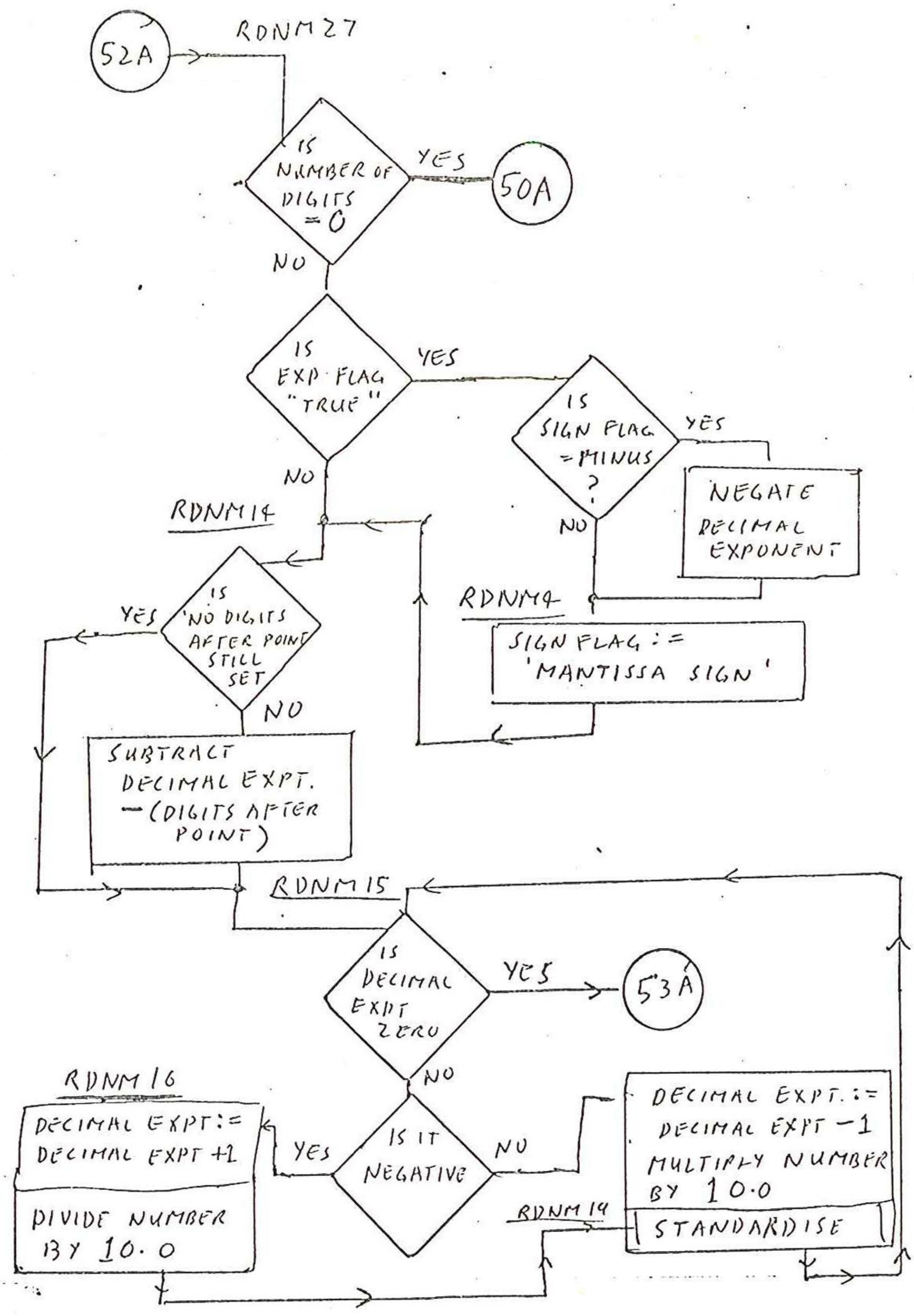
49N
'CHARACTER' := NSIGNF
IS NSIGNF 'SPACE' ?
YES
NO
49C
OTMX9
HAS SIGN BEEN OUTPUT ?
YES
NO
OUTPUT SIGNCH (SUBROUTINE OTCHA)
MARK 'SIGN OUTPUT'
OTMX10
OUTPUT 'CHARACTER' (OTCHA)
BEFPT := BEFPT −1
48A
(OTMX2)
49P
NSIGNF := 16 (CHARACTER ZERO)
HAS SIGN BEEN OUTPUT ?
YES
NO
OUTPUT SIGNCH (OTCHA)
MARK 'SIGN OUTPUT
OTMX5
OUTPUT '.' (DECIMAL POINT) OTCHA
48P

INPUT SUBROUTINE

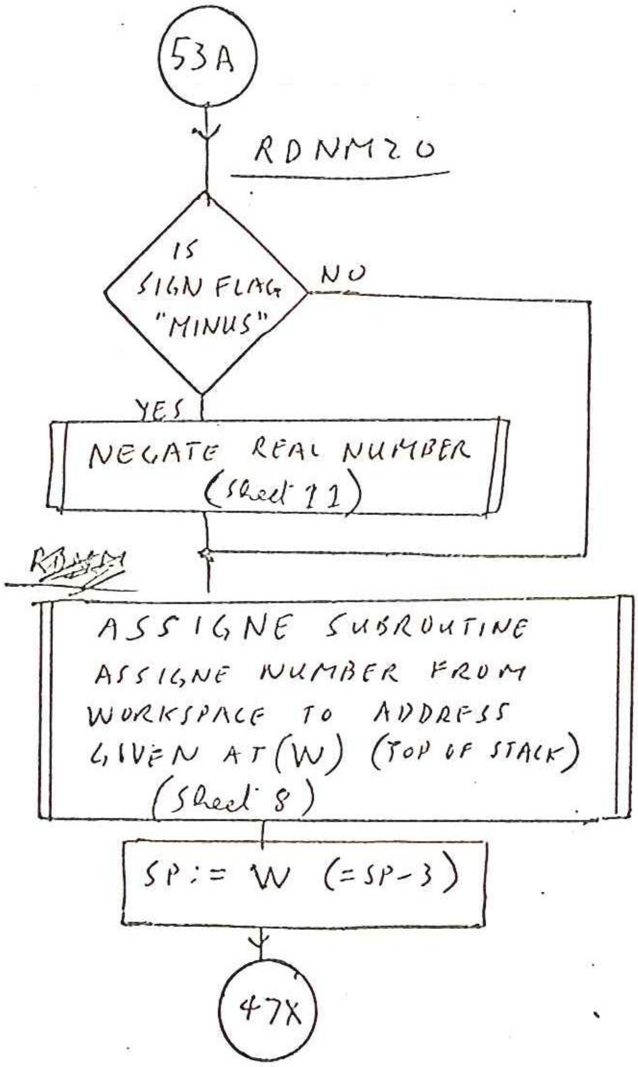


Sheet 51

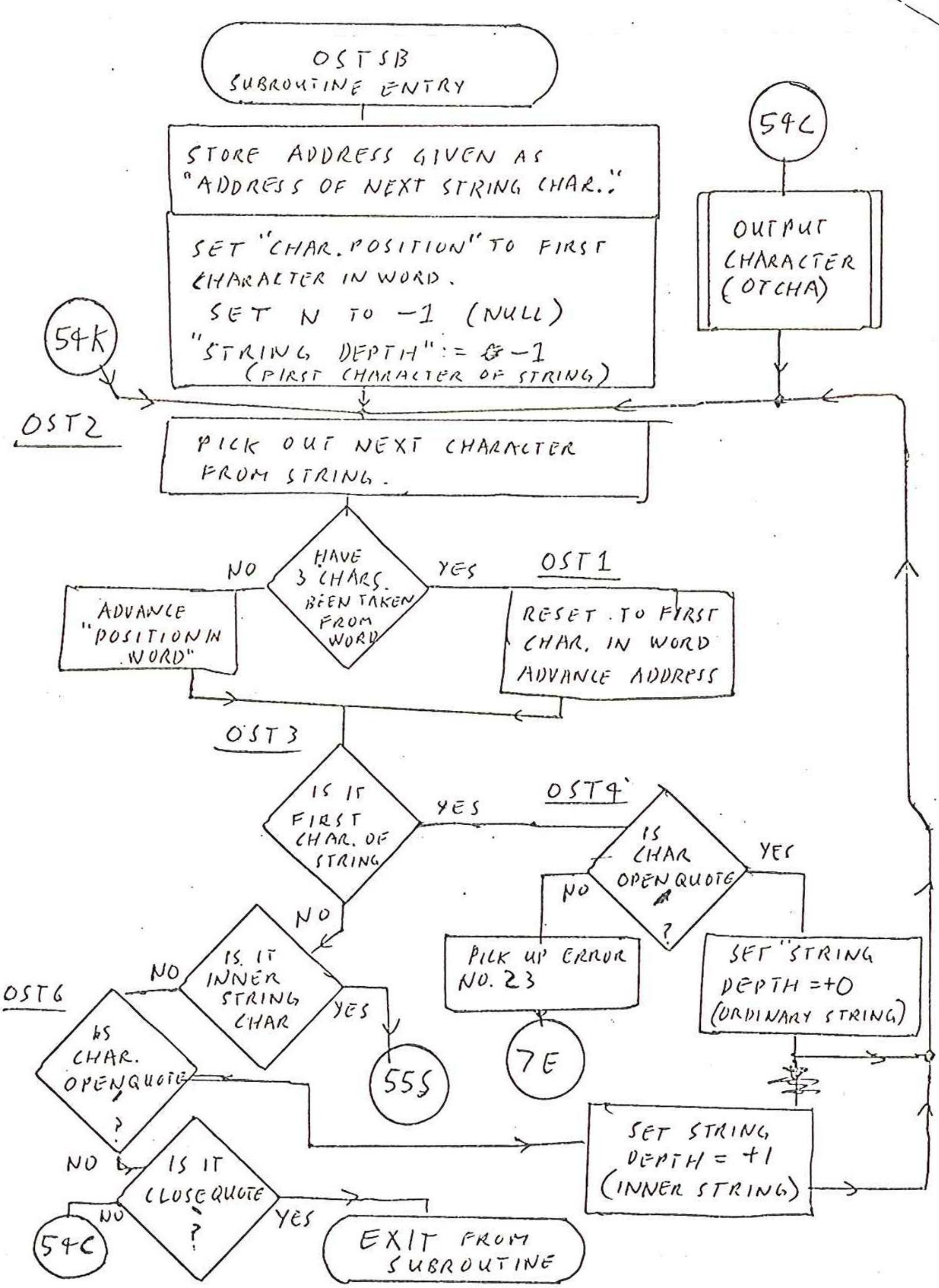

51C
RDNM5
SET REAL NUMBER FLAG "TRUE"
IS CHR. DECIMAL POINT ?
YES
NO
WAS IT ALREADY "TRUE" ?
RDNM6
SET DIGITS AFTER POINT "TRUE" AND = ZERO
YES
RDNM7
PICK UP ERROR NO. 18 (ILLEGAL FORM OF NUMBER)
51E
IS IT 'SUBSCRIPT 10' ?
YES
RDNM10
SIGN INPUT := "FALSE"
NO
IS EXP. FLAG "TRUE" ?
YES
NO
RDNM11
EXP. FLAG := "TRUE"
IS IT 'MINUS' ?
YES
RDNM8
SET SIGN BIT := "MINUS"
HAVE ANY DIGITS BEEN INPUT ?
NO
RDNM12
SET NUMBER := 1.0
SET DIGITS INPUT := 1
YES
IS IT 'PLUS' ?
YES
SIGN BIT := "PLUS"
RDNM9
RDNM13
STORE SIGN FLAG IN 'MANTISSA SIGN'
SIGN FLAG := "PLUS"
REAL NUMBER FLAG := "TRUE"
IS IT OPEN QUOTE ?
YES
NO
IS SIGN INPUT "TRUE"
YES
NO
RDNM26
PICK UP ERROR NO. 8
51E
SIGN FLAG := SIGN BIT
ERROR SUBROUTINE FAILEN
50E
52A
50A


52A
RDNM27
IS NUMBER OF DIGITS = 0
YES
50A
NO
IS EXP FLAG "TRUE"
YES
IS SIGN FLAG = MINUS ?
YES
NEGATE DECIMAL EXPONENT
NO
RDNM14
RDNM4
SIGN FLAG := 'MANTISSA SIGN'
IS 'NO DIGITS AFTER POINT' STILL SET
YES
NO
SUBTRACT DECIMAL EXPT. — (DIGITS AFTER POINT)
RDNM15
IS DECIMAL EXPT ZERO
YES
53A
NO
RDNM16
DECIMAL EXPT := DECIMAL EXPT +1
DIVIDE NUMBER BY 10.0
YES
IS IT NEGATIVE
NO
DECIMAL EXPT. := DECIMAL EXPT −1
MULTIPLY NUMBER BY 10.0
RDNM19
STANDARDISE

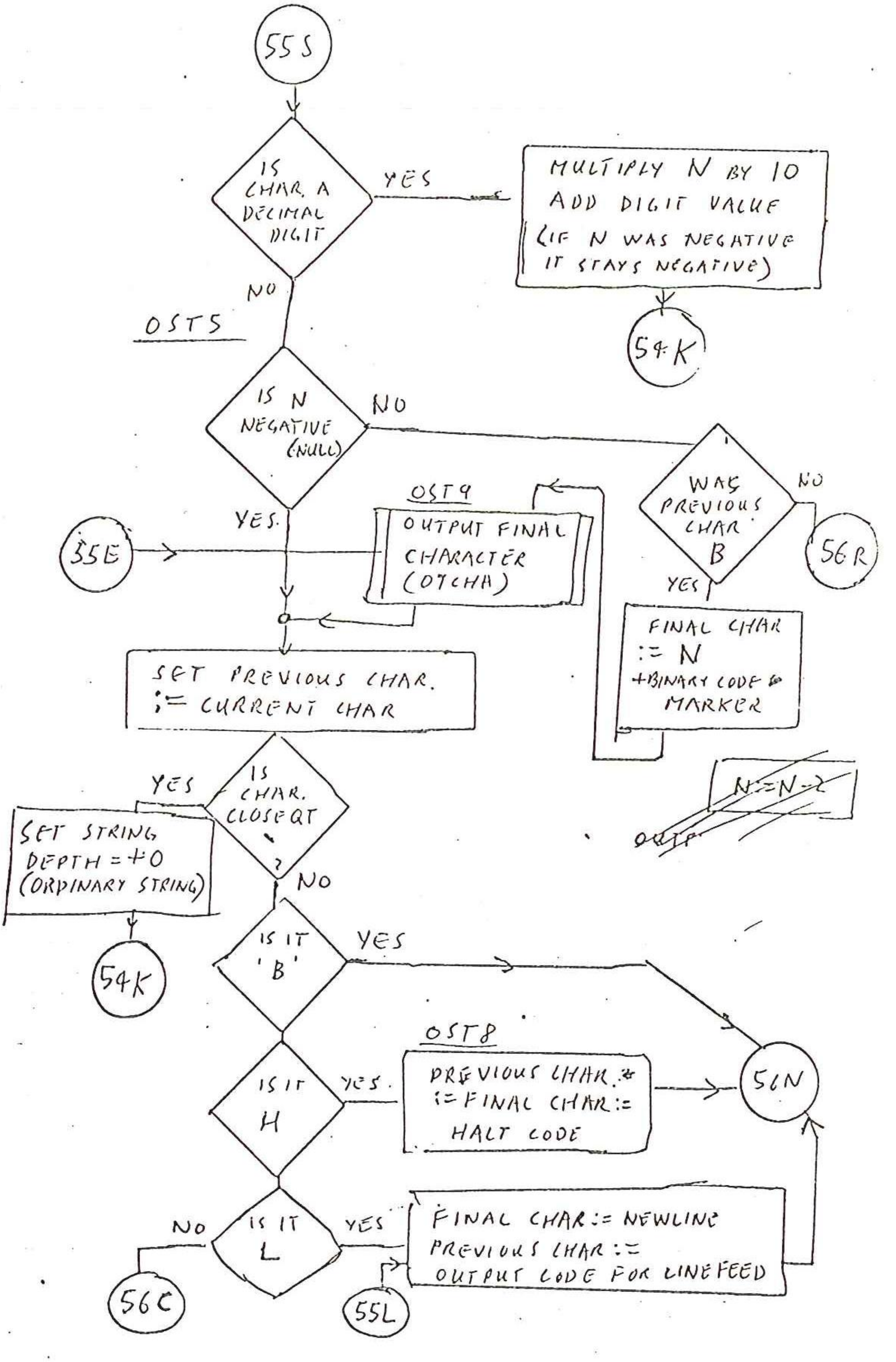
Sheet 53


53A
RDNM20
IS SIGN FLAG "MINUS"
NO
YES
NEGATE REAL NUMBER (Sheet 11)
ASSIGNE SUBROUTINE
ASSIGNE NUMBER FROM WORKSPACE TO ADDRESS GIVEN AT (W) (TOP OF STACK)
(Sheet 8)
SP := W (=SP-3)
47X

OUTPUT STRING SUBROUTINE.

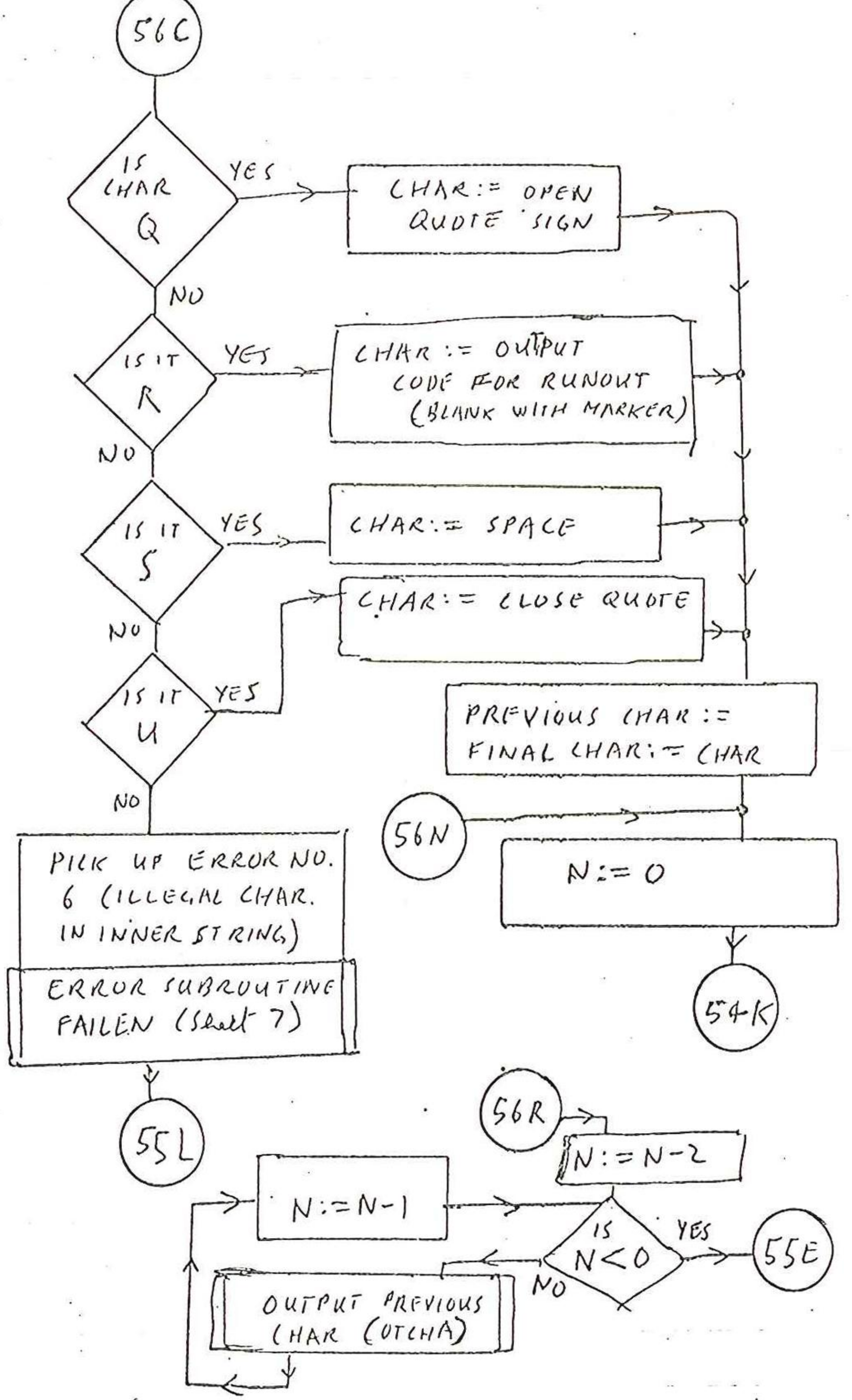


Sheet 55



55S
IS CHAR. A DECIMAL DIGIT
YES
MULTIPLY N BY 10
ADD DIGIT VALUE
(IF N WAS NEGATIVE IT STAYS NEGATIVE)
54K
NO
OST5
IS N NEGATIVE (NULL)
NO
WAS PREVIOUS CHAR B
NO
56R
YES
FINAL CHAR := N +BINARY CODE & MARKER
~~N:=N-2~~
~~OUT~~
YES.
OST9
55E
OUTPUT FINAL CHARACTER (OTCHA)
SET PREVIOUS CHAR. := CURRENT CHAR
IS CHAR. CLOSEQT ?
YES
SET STRING DEPTH = +0 (ORDINARY STRING)
54K
NO
IS IT 'B'
YES
OST8
IS IT H
YES.
PREVIOUS CHAR.* := FINAL CHAR := HALT CODE
56N
IS IT L
NO
56C
YES
FINAL CHAR := NEWLINE
PREVIOUS CHAR := OUTPUT CODE FOR LINEFEED
55L

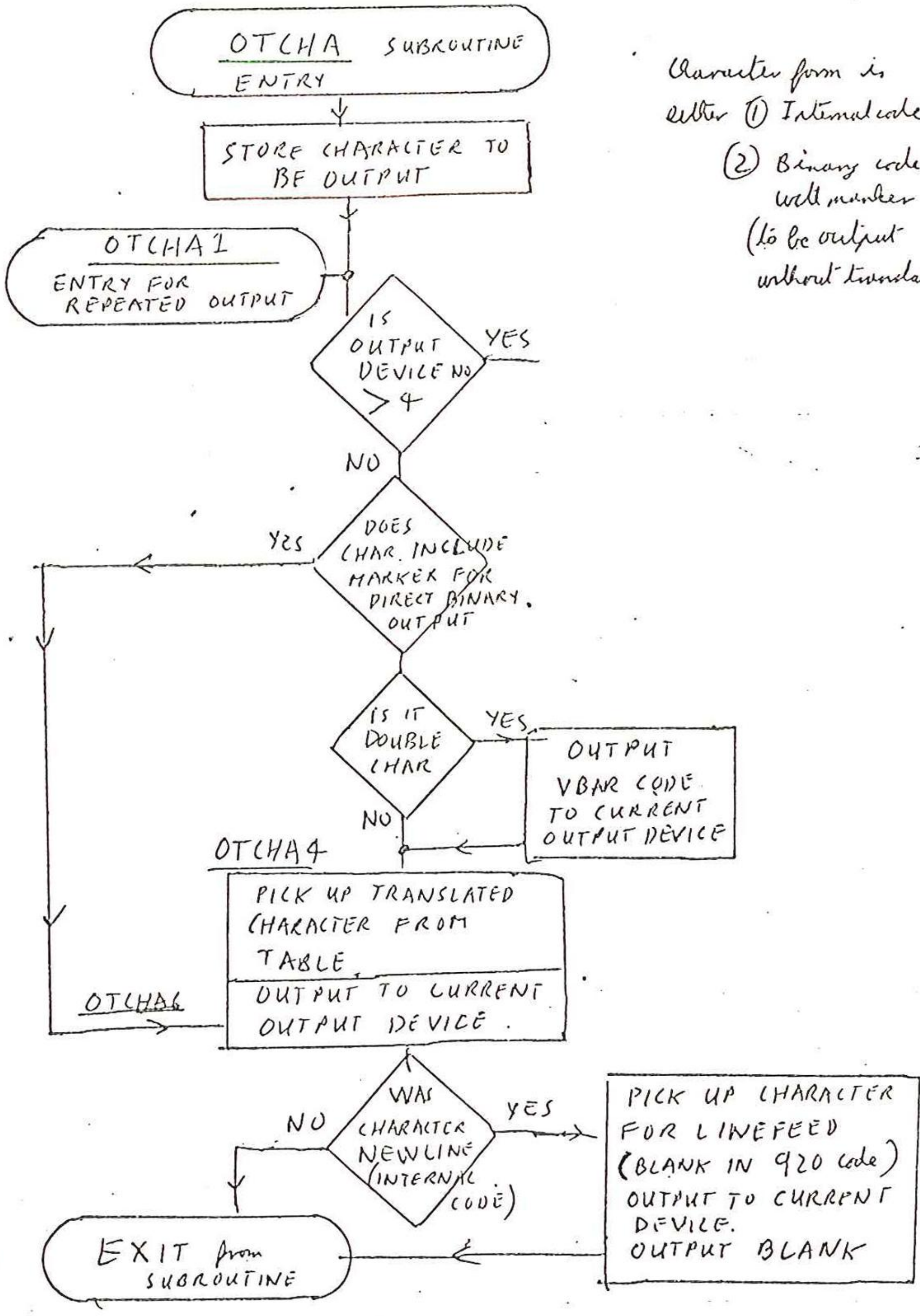
Sheet 56


56C
IS CHAR Q
YES
CHAR := OPEN QUOTE SIGN
NO
IS IT R
YES
CHAR := OUTPUT CODE FOR RUNOUT (BLANK WITH MARKER)
NO
IS IT S
YES
CHAR := SPACE
NO
IS IT U
YES
CHAR := CLOSE QUOTE
NO
PREVIOUS CHAR := FINAL CHAR := CHAR
56N
N := 0
54K
PICK UP ERROR NO. 6 (ILLEGAL CHAR. IN INNER STRING)
ERROR SUBROUTINE FAILEN (Sheet 7)
55L
56R
N := N-2
N := N-1
IS N<0
YES
55E
NO
OUTPUT PREVIOUS CHAR (OTCHA)

SUBROUTINE: OUTPUT ONE CHARACTER

(Common to 903 and 920 versions)

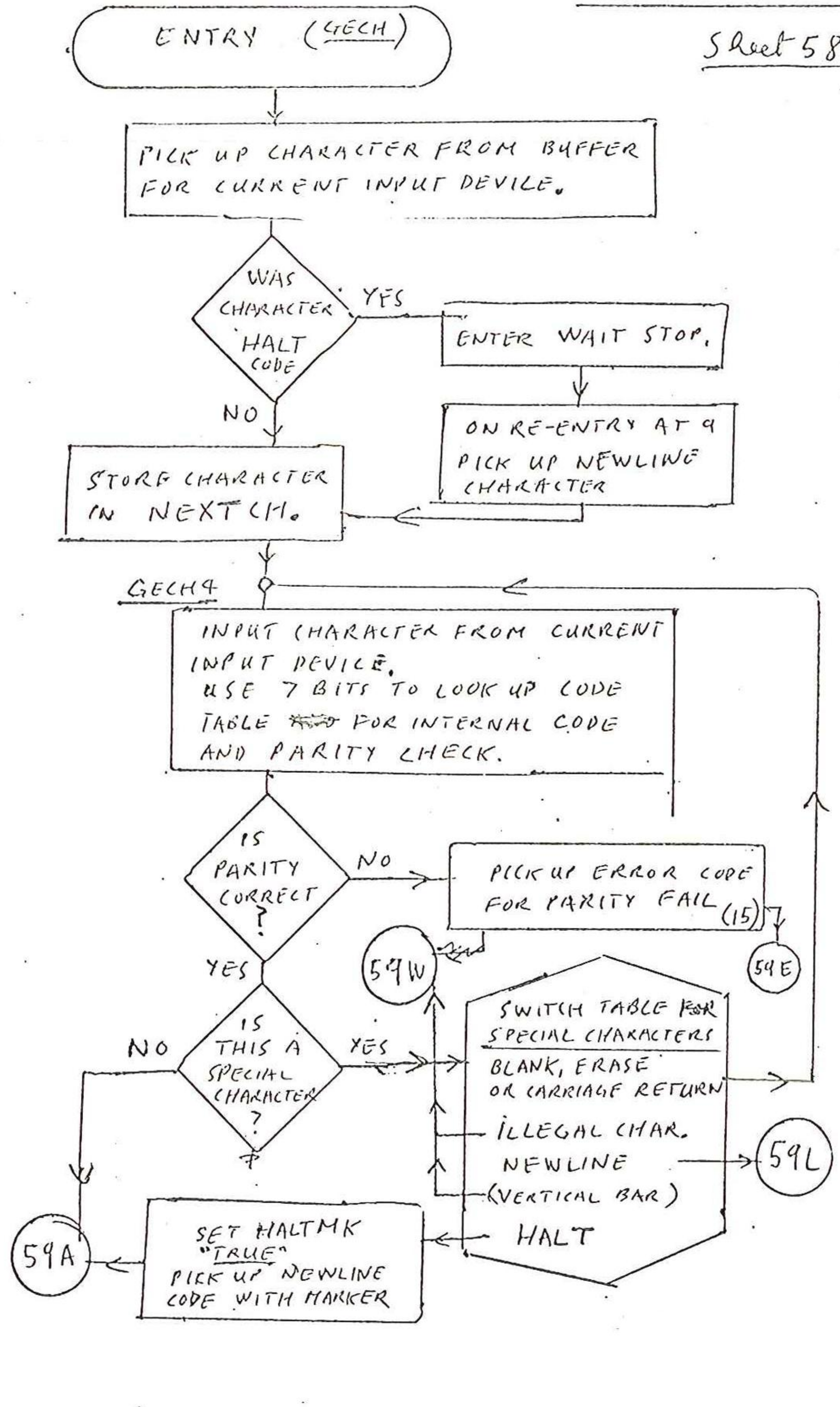
Character form is either (1) Internal code
(2) Binary code with marker (to be output without translation)

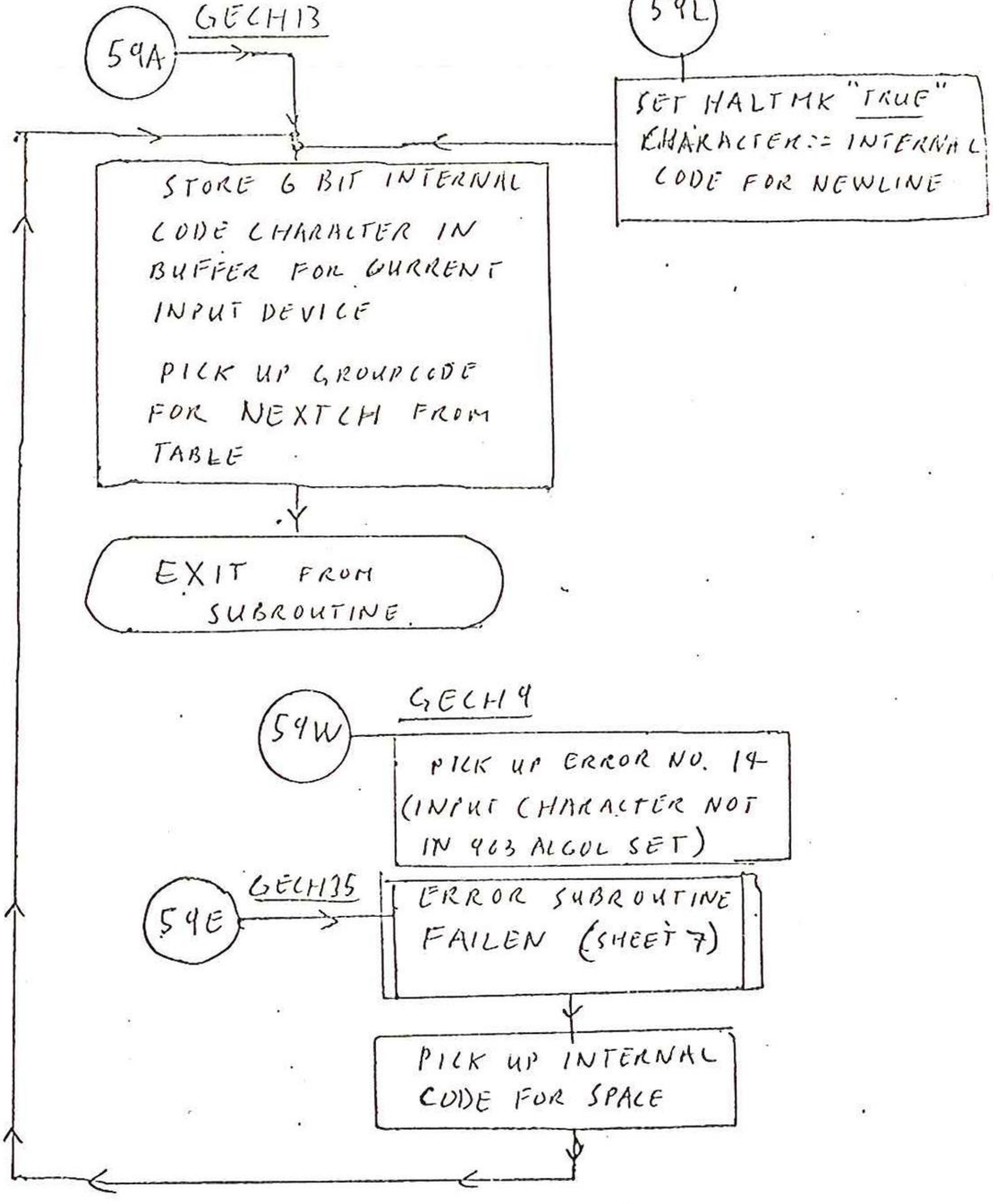


SUBROUTINE: GET NEXT CHARACTER

403 CODE VERSION

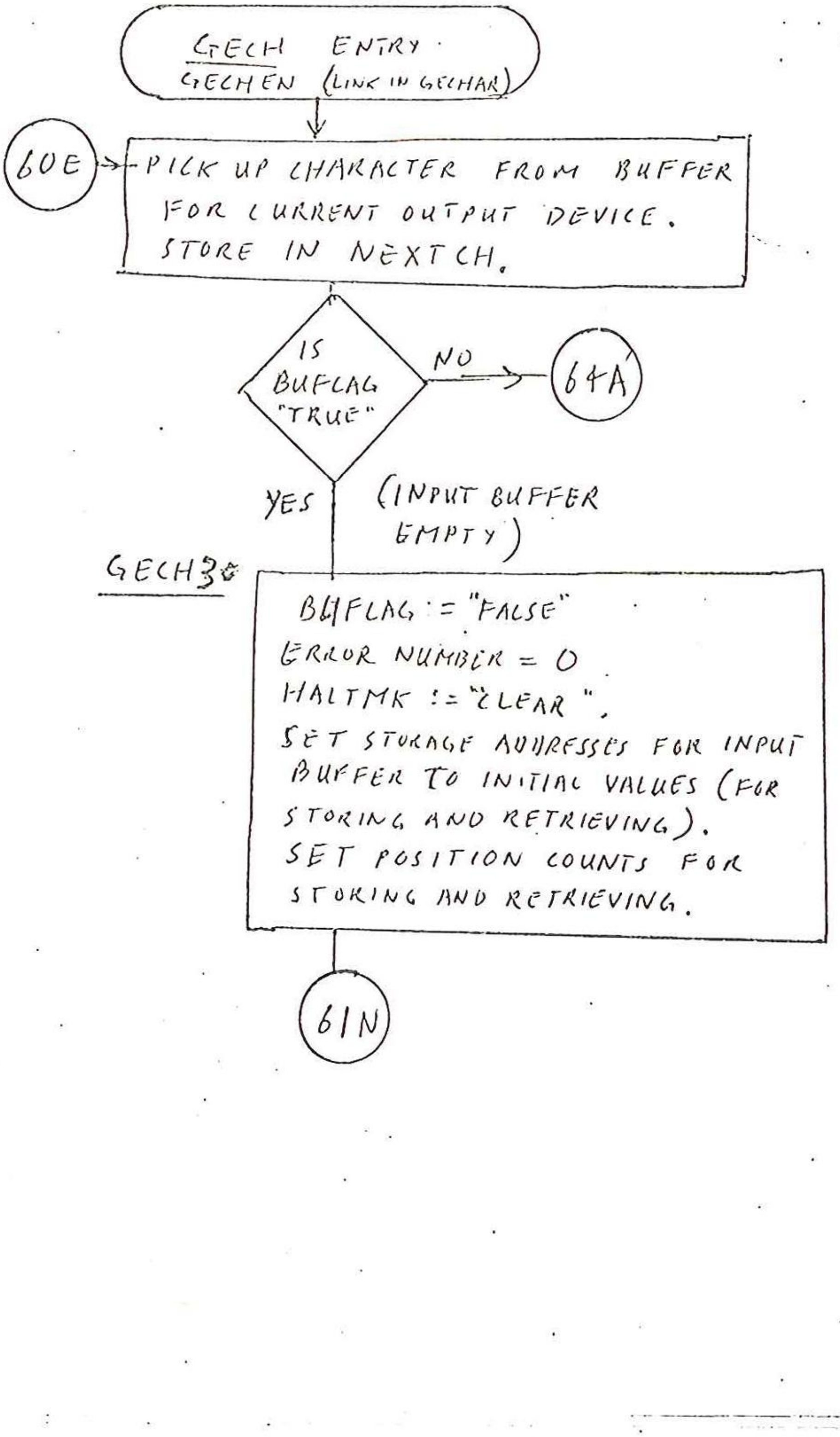
Sheet 58

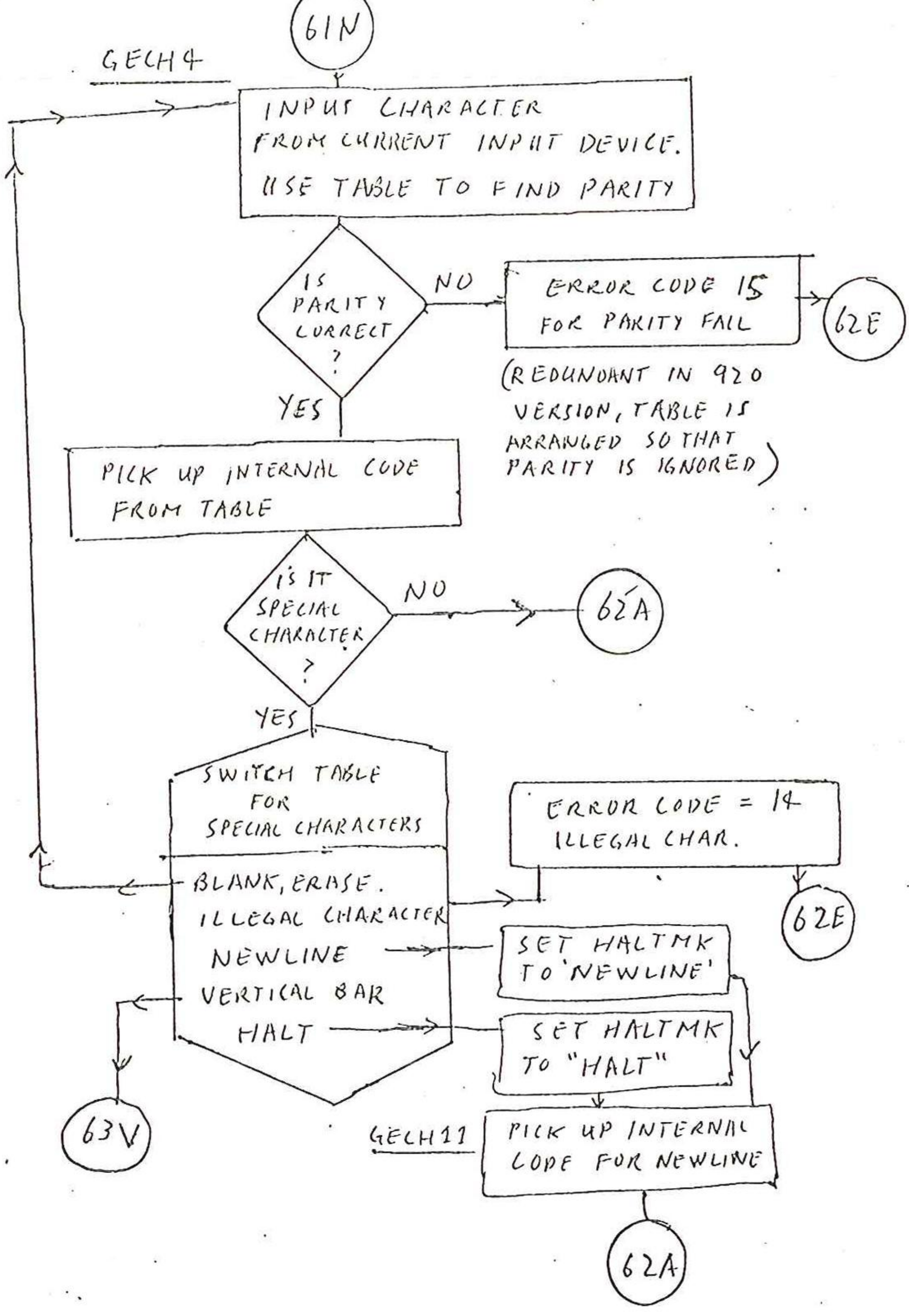



GECH13
54A
54L
SET HALTMK "TRUE"
CHARACTER := INTERNAL CODE FOR NEWLINE
STORE 6 BIT INTERNAL CODE CHARACTER IN BUFFER FOR CURRENT INPUT DEVICE
PICK UP GROUPCODE FOR NEXTCH FROM TABLE
EXIT FROM SUBROUTINE.
GECH4
54W
PICK UP ERROR NO. 14 (INPUT CHARACTER NOT IN 903 ALGOL SET)
GECH35
54E
ERROR SUBROUTINE FAILEN (SHEET 7)
PICK UP INTERNAL CODE FOR SPACE

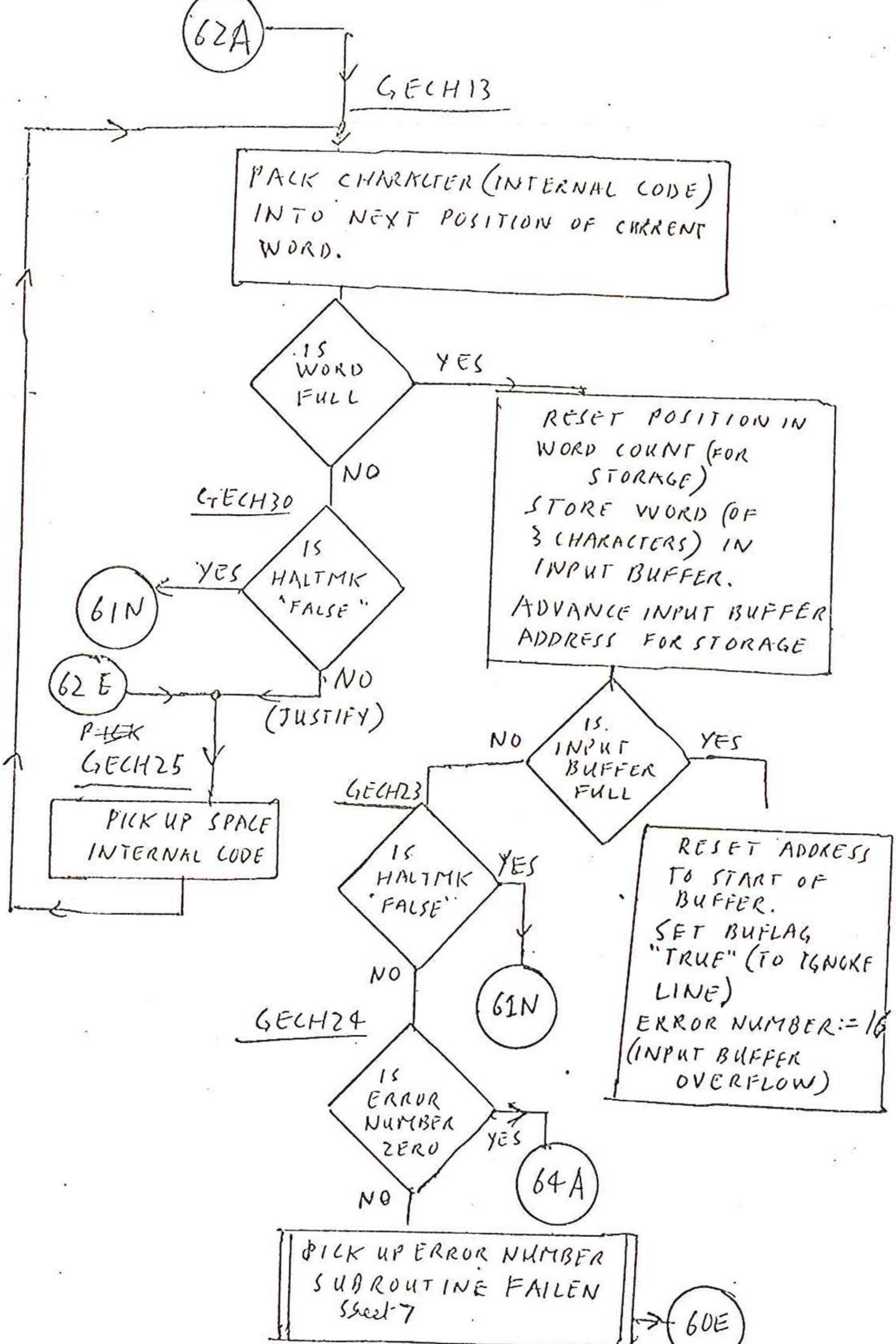
SUBROUTINE: GET NEXT CHARACTER

(920 CODE VERSION)

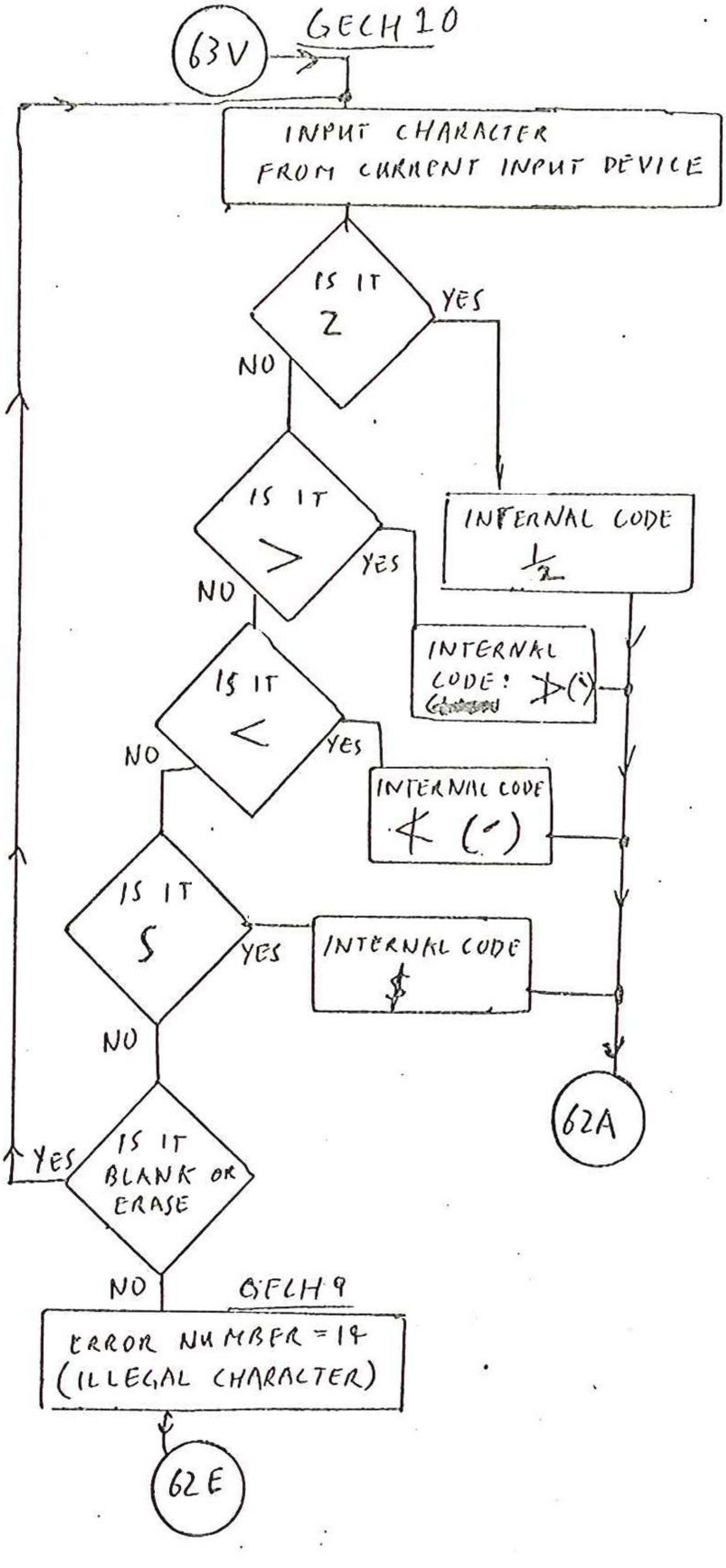



61N
GECH4
INPUT CHARACTER FROM CURRENT INPUT DEVICE.
USE TABLE TO FIND PARITY
IS PARITY CORRECT ?
NO
ERROR CODE 15 FOR PARITY FAIL
62E
(REDUNDANT IN 920 VERSION, TABLE IS ARRANGED SO THAT PARITY IS IGNORED)
YES
PICK UP INTERNAL CODE FROM TABLE
IS IT SPECIAL CHARACTER ?
NO
62A
YES
SWITCH TABLE FOR SPECIAL CHARACTERS
BLANK, ERASE.
ILLEGAL CHARACTER
NEWLINE
VERTICAL BAR
HALT
ERROR CODE = 14 ILLEGAL CHAR.
62E
SET HALTMK TO 'NEWLINE'
SET HALTMK TO "HALT"
63V
GECH11
PICK UP INTERNAL CODE FOR NEWLINE
62A

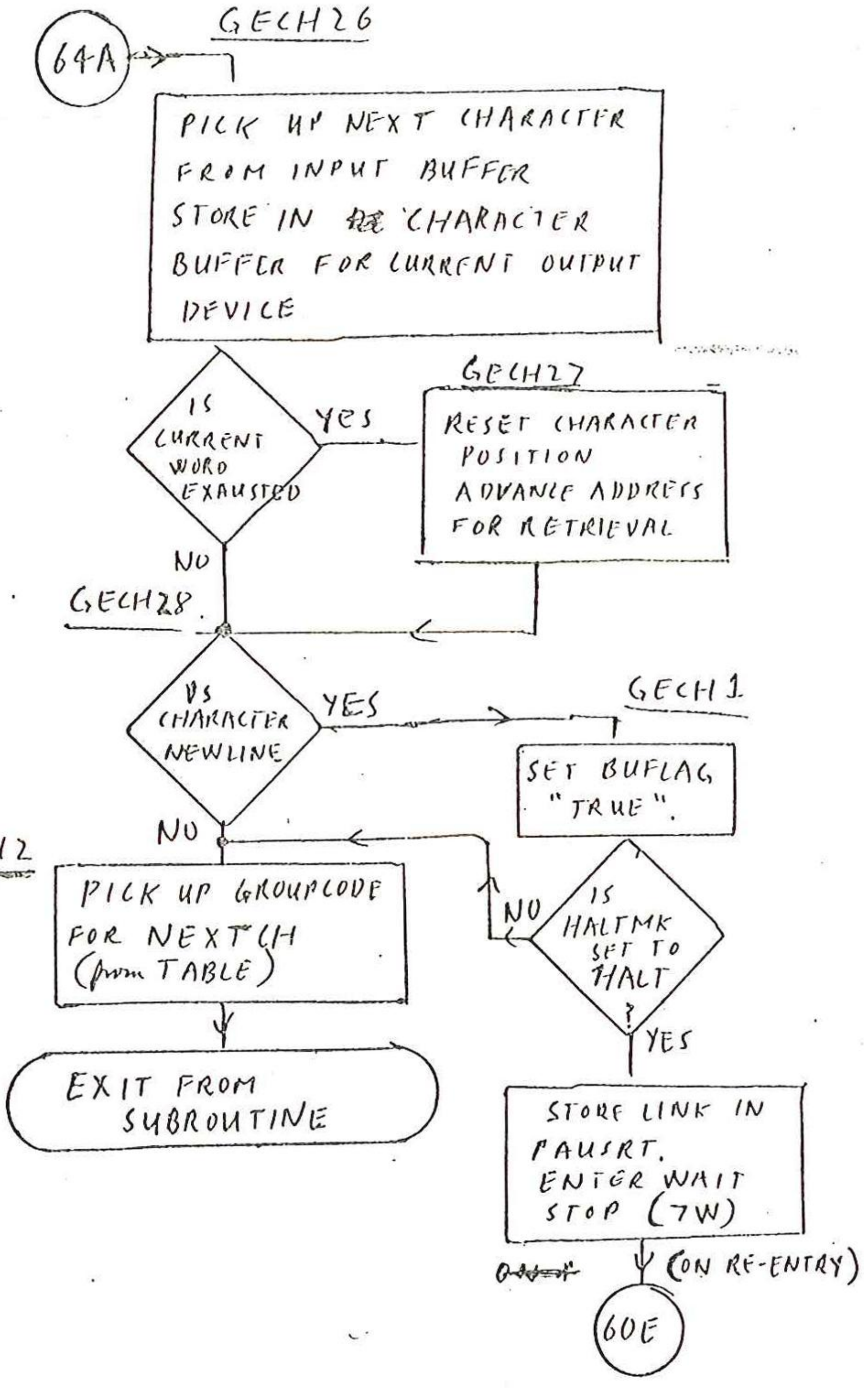
Sheet 62


62A
GECH13
PACK CHARACTER (INTERNAL CODE) INTO NEXT POSITION OF CURRENT WORD.
IS WORD FULL
YES
NO
RESET POSITION IN WORD COUNT (FOR STORAGE)
STORE WORD (OF 3 CHARACTERS) IN INPUT BUFFER.
ADVANCE INPUT BUFFER ADDRESS FOR STORAGE
GECH30
IS HALTMK "FALSE"
YES
61N
NO
62 E
(JUSTIFY)
~~PICK~~
GECH25
PICK UP SPACE INTERNAL CODE
NO
IS INPUT BUFFER FULL
YES
GECH23
IS HALTMK "FALSE"
YES
NO
61N
RESET ADDRESS TO START OF BUFFER.
SET BUFLAG "TRUE" (TO IGNORE LINE)
ERROR NUMBER:=16
(INPUT BUFFER OVERFLOW)
GECH24
IS ERROR NUMBER ZERO
YES
64A
NO
PICK UP ERROR NUMBER
SUBROUTINE FAILEN
Sheet 7
60E

Sheet 63



63V
GECH10
INPUT CHARACTER FROM CURRENT INPUT DEVICE
IS IT Z
YES
NO
INTERNAL CODE ½
IS IT >
YES
NO
INTERNAL CODE: ≯(')
IS IT <
YES
NO
INTERNAL CODE ≮(')
IS IT S
YES
NO
INTERNAL CODE $
62A
IS IT BLANK OR ERASE
YES
NO
GECH9
ERROR NUMBER = 14 (ILLEGAL CHARACTER)
62E


64A
GECH26
PICK UP NEXT CHARACTER FROM INPUT BUFFER STORE IN ~~RE~~ CHARACTER BUFFER FOR CURRENT OUTPUT DEVICE
IS CURRENT WORD EXAUSTED
YES
GECH27
RESET CHARACTER POSITION ADVANCE ADDRESS FOR RETRIEVAL
NO
GECH28
IS CHARACTER NEWLINE
YES
GECH1
SET BUFLAG "TRUE".
NO
GECH2
PICK UP GROUPCODE FOR NEXTCH (from TABLE)
IS HALTMK SET TO HALT ?
NO
YES
EXIT FROM SUBROUTINE
STORE LINK IN PAUSRT. ENTER WAIT STOP (7W)
(ON RE-ENTRY)
60E

ALGOL LOADER: OVERALL FLOW