S C A L P

BLOCK DIAGRAMS

CCM-8A

Stephen J. Garland
Anthony W. Knapp
Thomas E. Kurtz

15 May 1963

Computation Center
Dartmouth College

Introduction

SCALP is an adaptation of ALGOL-30 for load-and-go operation. ALGOL-30 is a partial implementation of ALGOL-60 for the LGP-30, but which has not been publically released.

SCALP places these restrictions on ALGOL: (1) No blocks or procedures, except special functions; (2) No Boolean variables or operators; (3) No conditional expressions; (4) Only the while type for-statement; (5) No nested switch declarations, though nested switch calls are permitted; (6) Special integer divide not included. There are several symbol transliterations, and a number of machine-imposed limitations, but in all other general respects SCALP adheres to the form and spirit of ALGOL.

The reader is referred to CCM-7 (A Manual for SCALP) for detailed information on the external characteristics of SCALP and its operation.

This memorandum contains the block diagrams for SCALP. The coding sheets appear in a companion memorandum, CCM-8B, while the library of special functions is documented in CCM-8C.

In the absence of a detailed discussion of the block diagrams, it should be noted that the compiling follows closely the last-in, first-out method of translation proposed by Samelson and Bauer, and others. The symbol table follows the suggestion of E. J. Williams. These methods give extremely fast compiling speeds.

The principal work on SCALP was done by two students, Stephen J. Garland and Anthony W. Knapp. Garland did the compiler while Knapp composed the interpreter. Their work rests on the earlier work of ALGOL-30 which included also the efforts of Robert Hargraves and Jorge Llacer.

SCALP Instructions

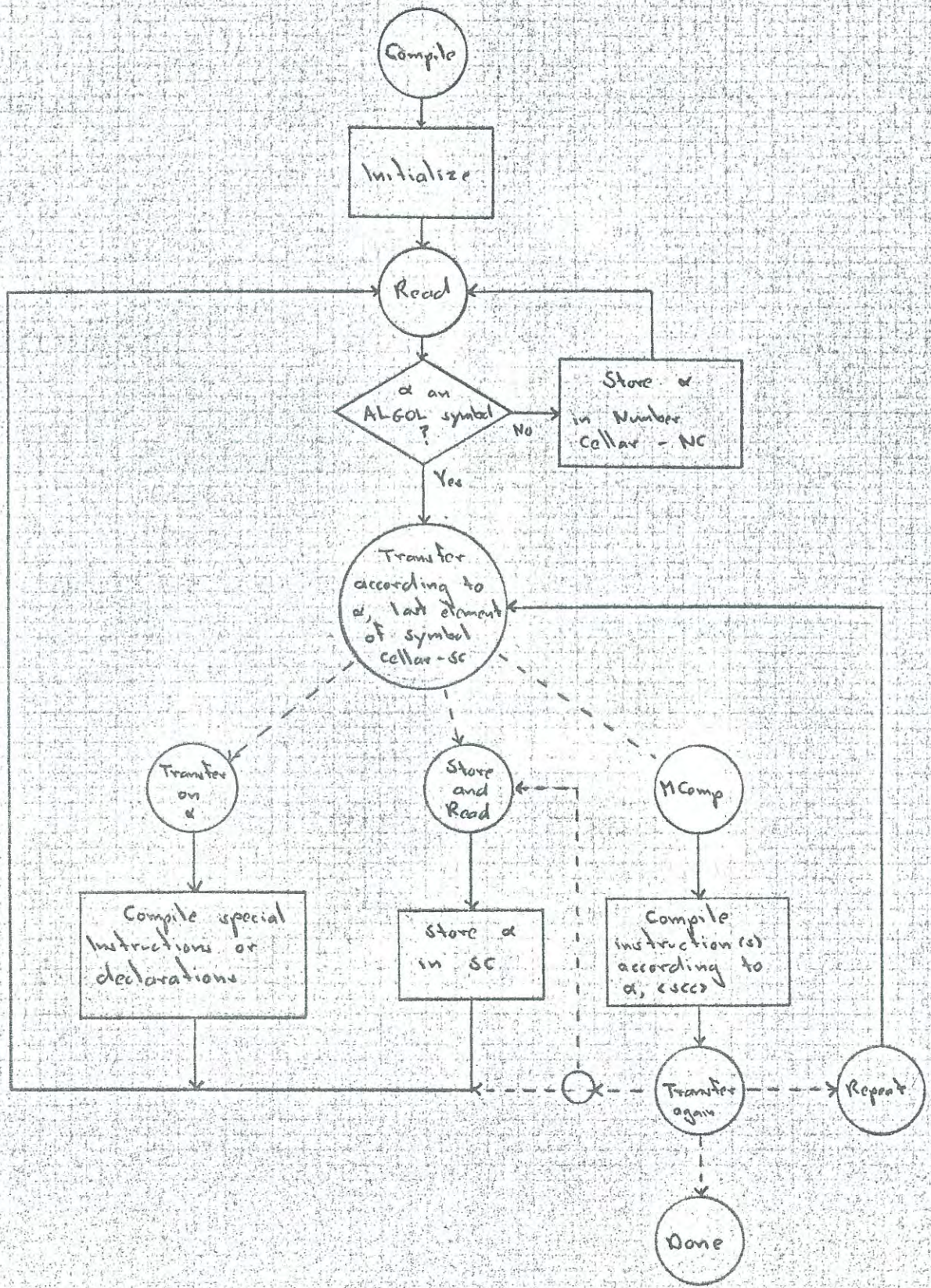
Inst. \ Tag	0	4	8	J
Z	abs	sign	cos *	sine *
B	Bring			
Y	Subtract From		No operation	Subtract From
R	random *		sqrt *	Relational
I		Divide Into		Divide Into
O		Divide	Change sign	Divide
N	Multiply			Multiply
M	Matrix	Read Integer	Read Real	
P	Print			Label
E	* Power *		entier *	Power *
U	Transfer	Switch	Exit	
T	Title	c.r.	tab.	
H	Trace Integer	Hold Integer	Hold Real	Trace Real
C	Float	Round		ln *
A	Add		arctan *	Add
S	Subtract		exp *	Subtract

* on library tape.

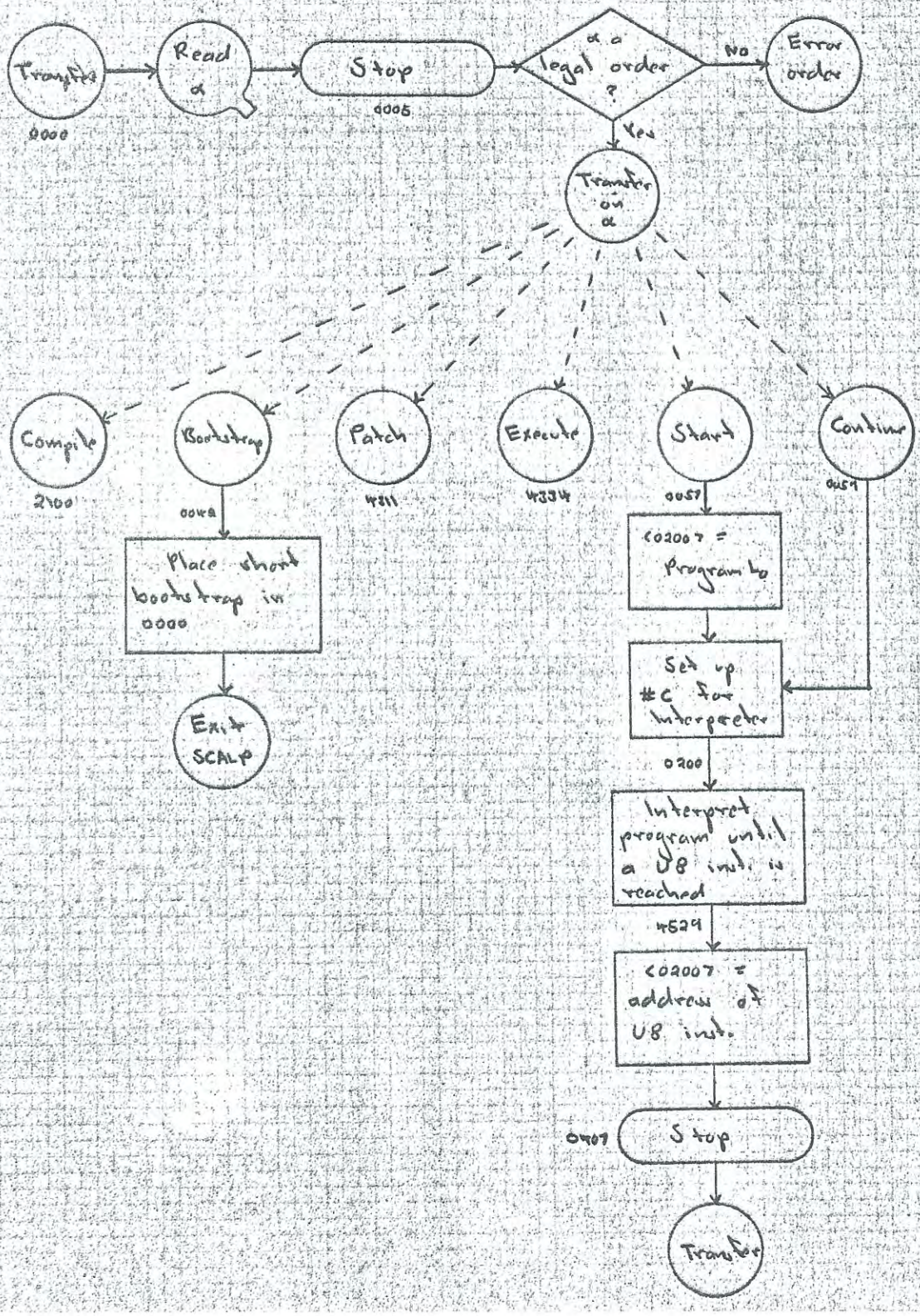
BY_____ DATE_____ SUBJECT_____ SHEET NO. 1 OF ____

CHKD. BY_____ DATE_____ JOB NO._____

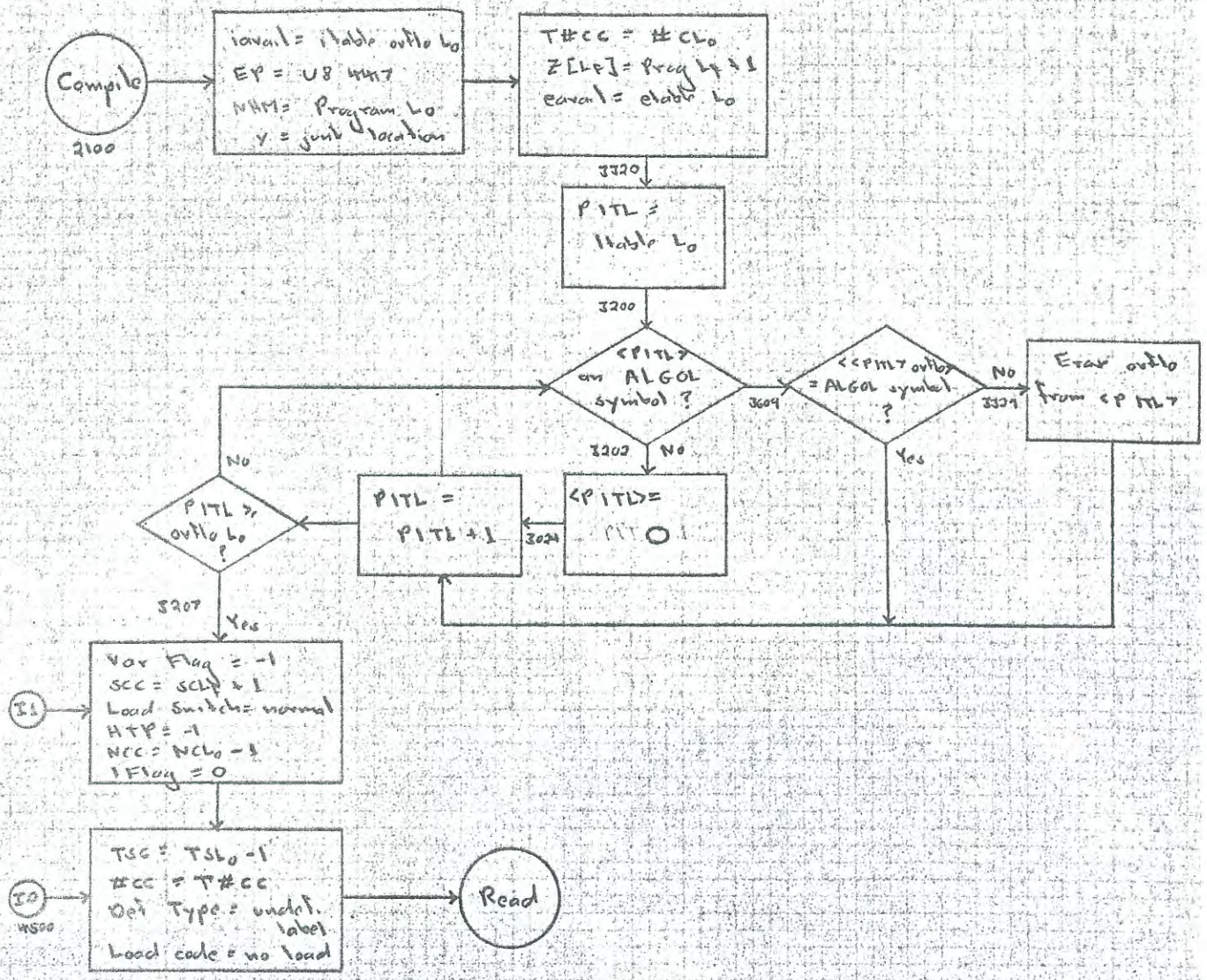
General Flow Diagram



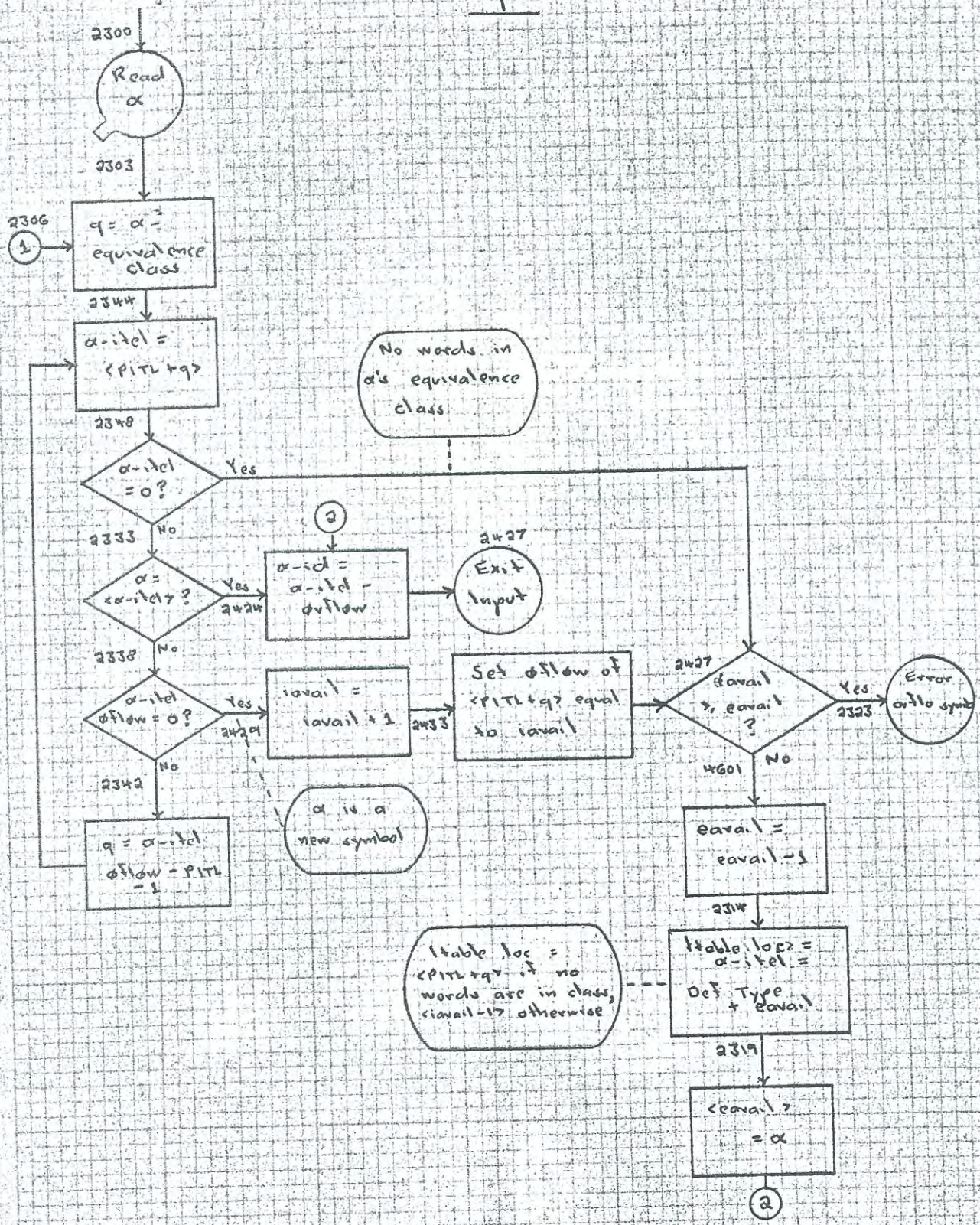
Transfer


Transfer
0000
Read α
Stop
0005
α a legal order ?
No
Error order
Yes
Transfer on α
Compile
2100
Bootstrap
00ea
Patch
4211
Execute
4334
Start
0057
Continue
0059
Place short bootstrap in 0000
Exit SCALP
<02007 = Program b0
Set up #C for Interpreter
0200
Interpret program until a UB inst. is reached
4529
<02007 = address of UB inst.
0407
Stop
Transfer

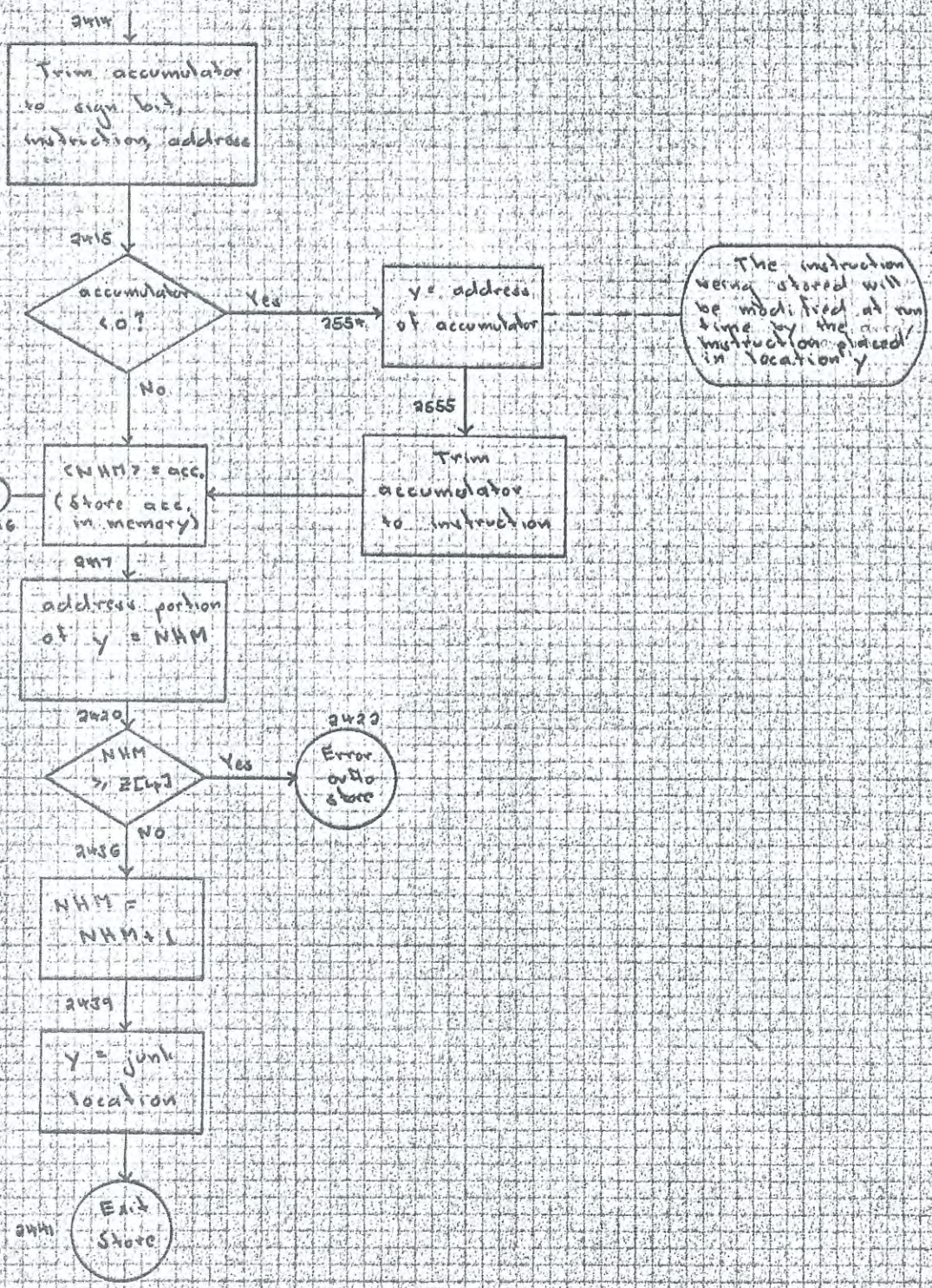
Initialization



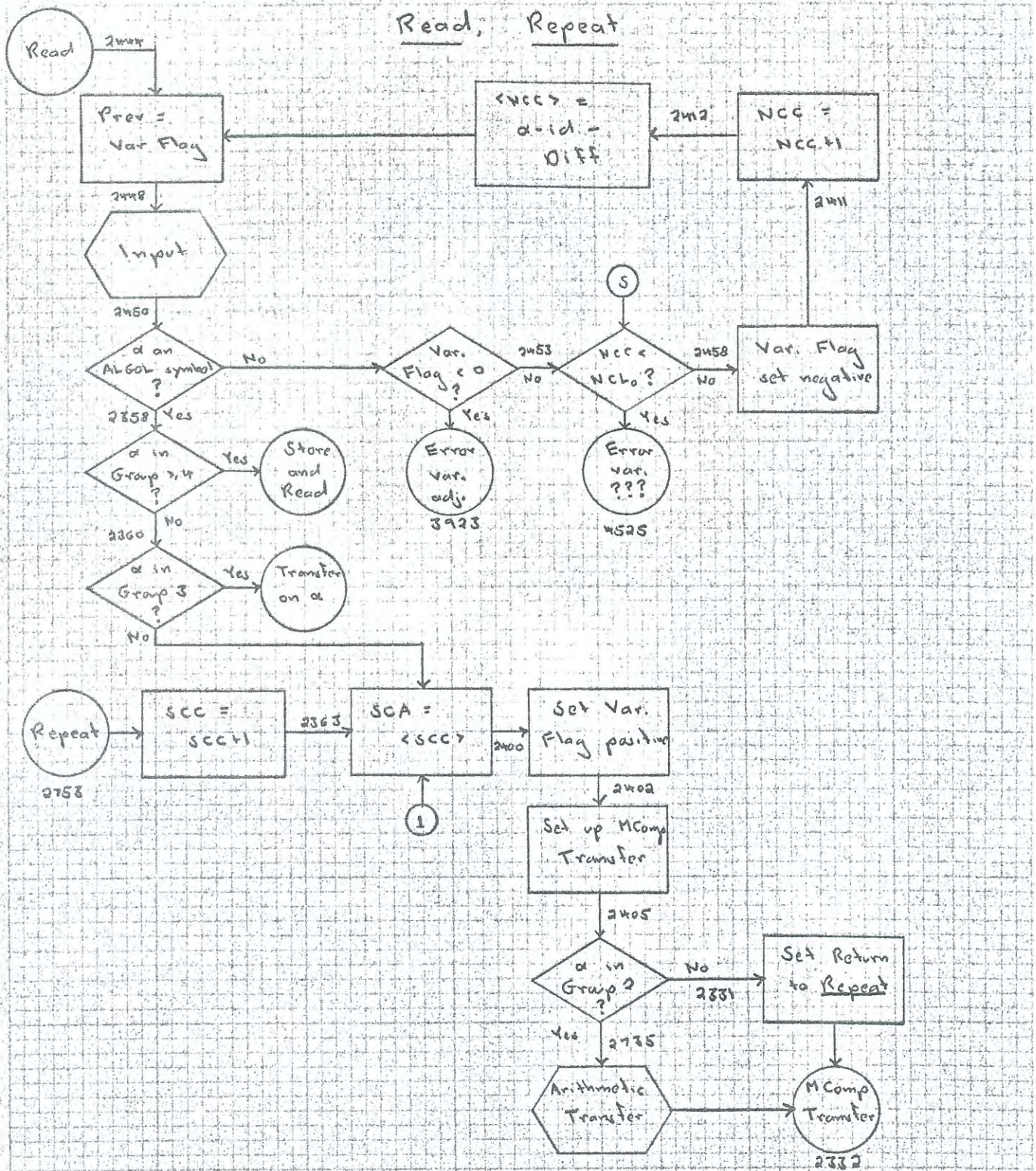
Input


2300
Read α
2303
2306
1
q = α-1 equivalence class
2344
α-itel = <PITL+q>
2348
α-itel = 0?
Yes
No
No words in α's equivalence class
2333
α = <α-itel>?
Yes
2424
α-itel = α-itel - ofllow
2427
Exit Input
2338
No
α-itel ofllow = 0?
Yes
2424
iavail = iavail + 1
2433
Set ofllow of <PITL+q> equal to iavail
2427
iavail > eavail ?
Yes
2323
Error ofllo symb
No
4601
eavail = eavail - 1
2342
q = α-itel ofllow = PITL - 1
α is a new symbol
2314
Itable loc = α-itel = Def Type + eavail
Itable loc = <PITL+q> if no words are in class, <iavail-1> otherwise
2319
<eavail> = α
2
2

Store


2414
Trim accumulator to sign bit, instruction, address
2415
accumulator < 0 ?
Yes
2554
y = address of accumulator
The instruction word stored will be modified at run time by the instruction placed in location y
No
2555
Trim accumulator to instruction
(1)
2416
(NHM) = acc. (store acc. in memory)
2417
address portion of y = NHM
2420
NHM ≥ Z[4]
Yes
2423
Error out to store
No
2436
NHM = NHM + 1
2439
y = junk location
2441
Exit Store

Read, Repeat

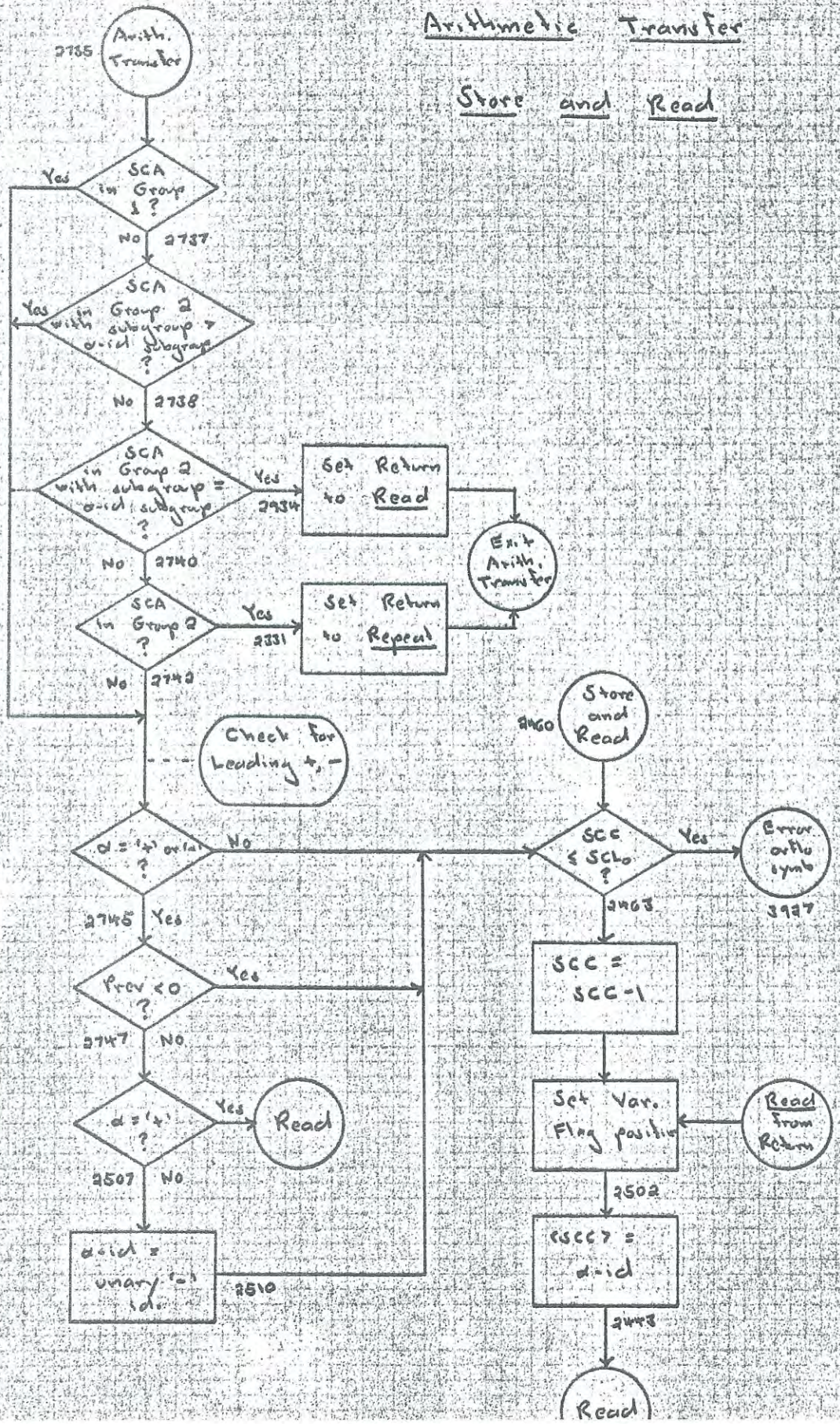

Read
2444
Prev = Var Flag
2448
Input
2450
α an ALGOL symbol ?
No
Yes
2458
α in Group 2, 4 ?
Yes
Store and Read
No
2460
α in Group 3 ?
Yes
Transfer on α
No
Var. Flag < 0 ?
No
Yes
Error var. adj.
3923
2453
Ncc < NCL0 ?
No
Yes
Error var. ???
4525
2458
Var. Flag set negative
S
2411
Ncc = Ncc+1
2412
<Ncc> = α-id - Diff
Repeat
2758
SCC = SCC+1
2363
SCA = <SCC>
2400
Set Var. Flag positive
2402
Set up MComp Transfer
2405
α in Group 2 ?
No
2331
Set Return to Repeat
Yes
2735
Arithmetic Transfer
MComp Transfer
2332
1

BY_____ DATE_____ SUBJECT_____ SHEET NO. 7 OF_____

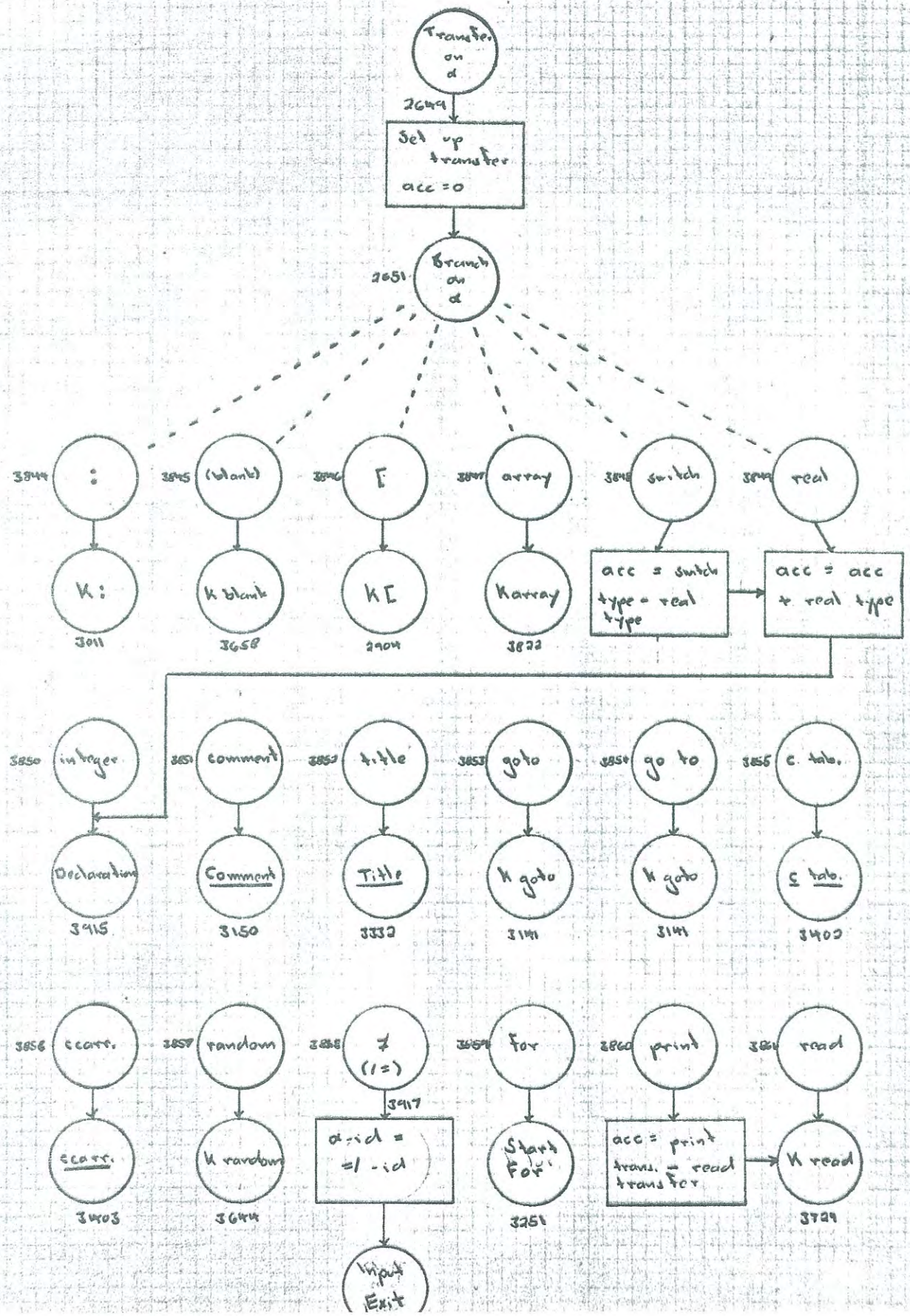
CHKD. BY_____ DATE_____ JOB NO._____

Arithmetic Transfer

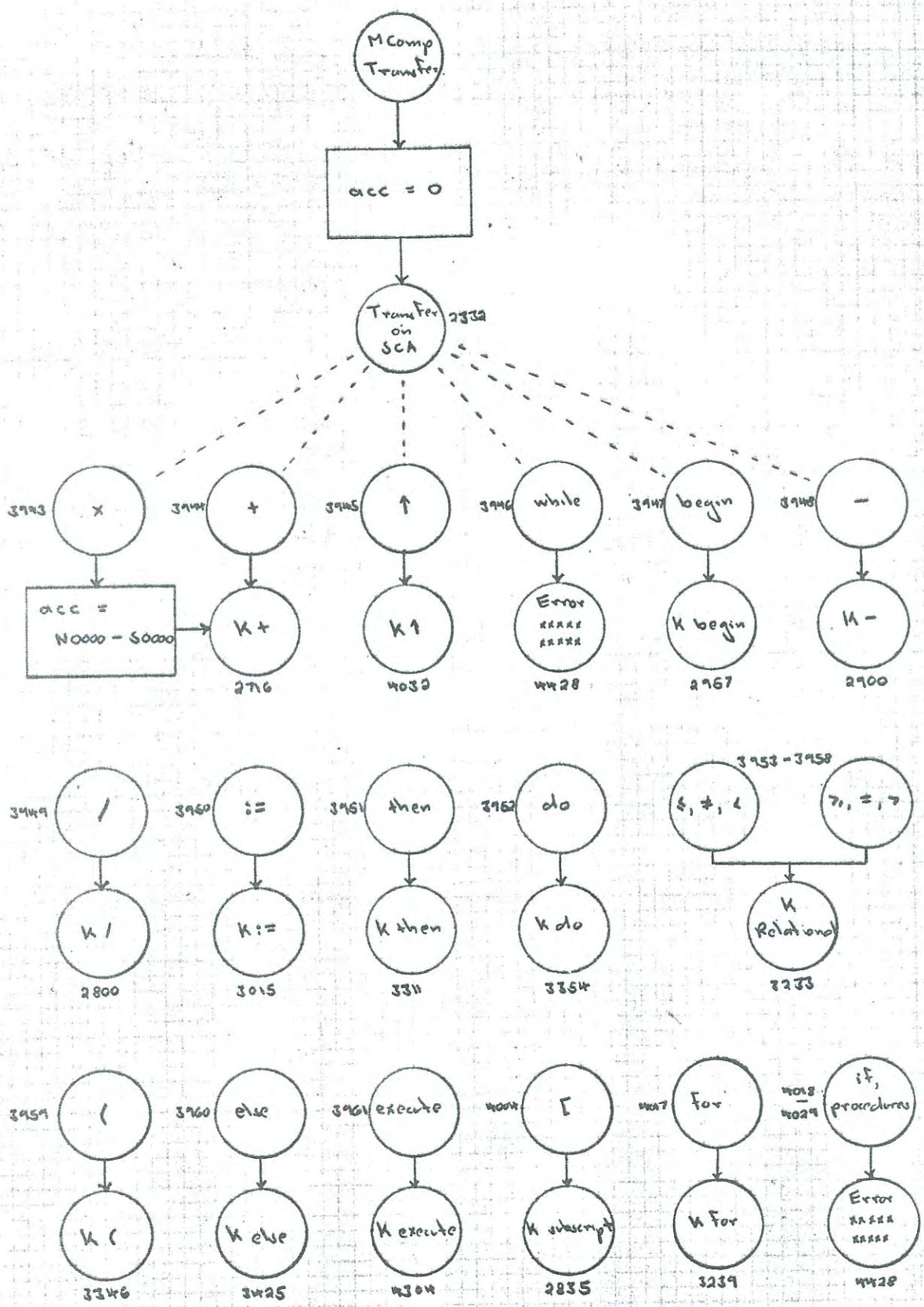
Store and Read



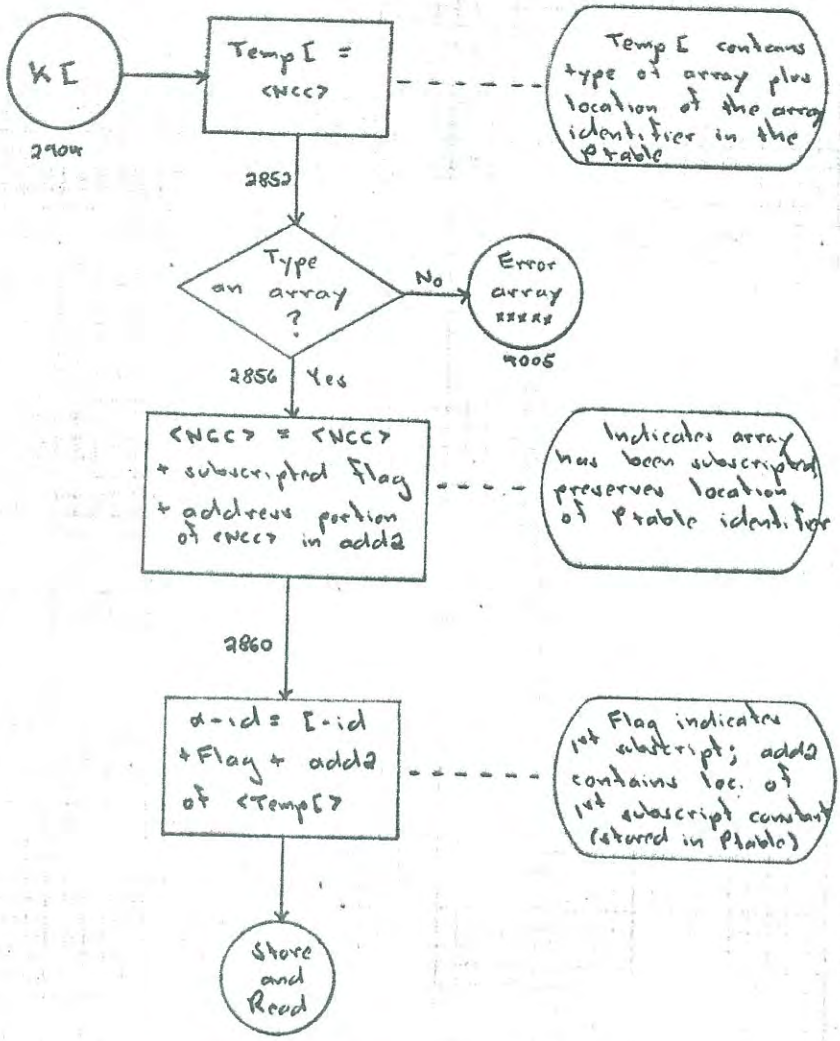
Transfer on α



MComp Transfer



Array Headings



For A[i,j], the following is compiled:

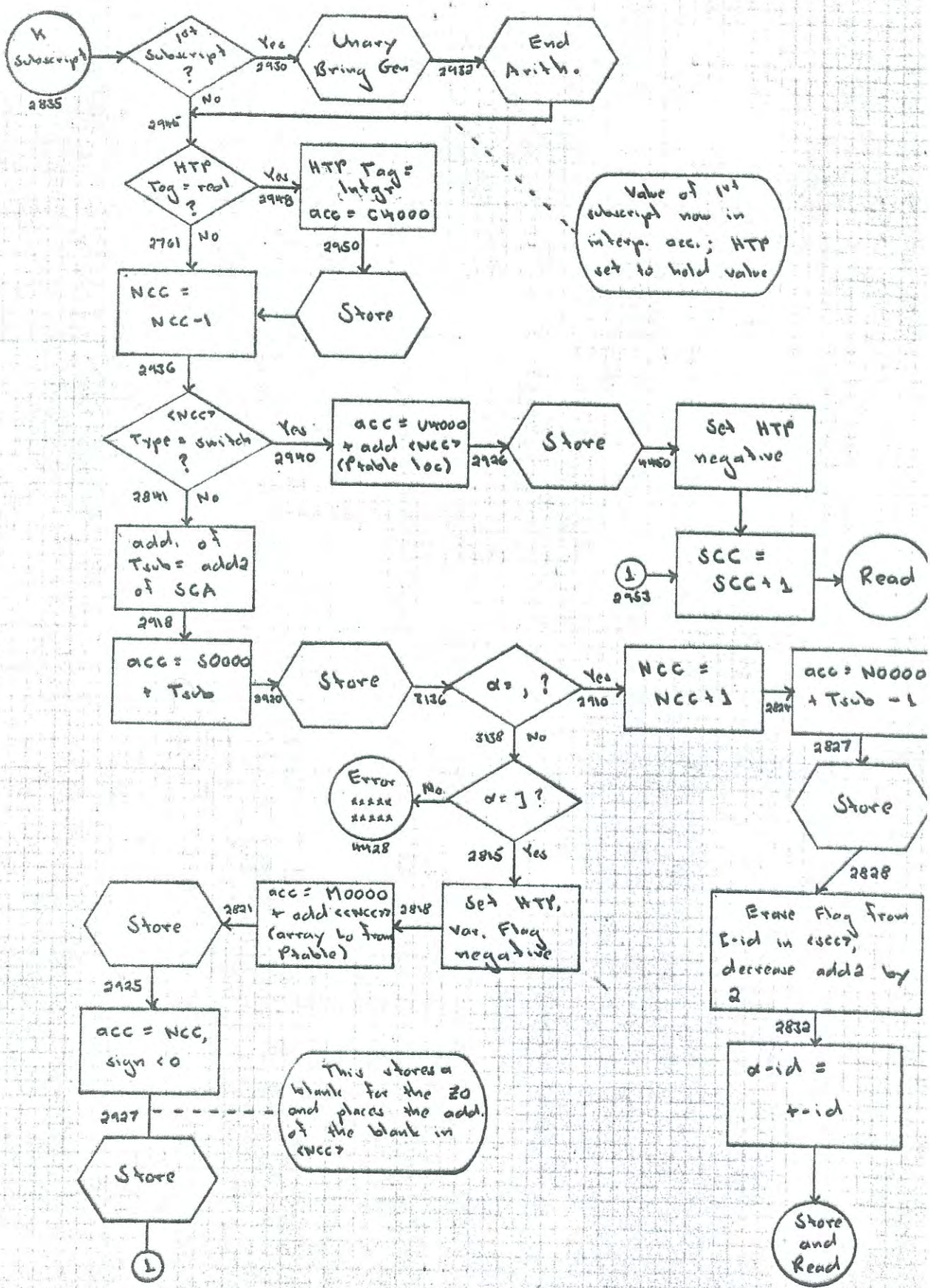
BO i

Compiled when a=, , SCA = [;
SO lower bound of 1st subscript
NO dimension of 2nd subscript
} Addresses computed from add2 of [-id

[-id stored to compile - AO j

Compiled when a=], SCA = [
SO lower bound of 2nd subscript
MO 1st loc of array - address stored in Ptable
ZO location to store address computed

K[sets up all constants for this compilation;
]A subscript generates the instructions.

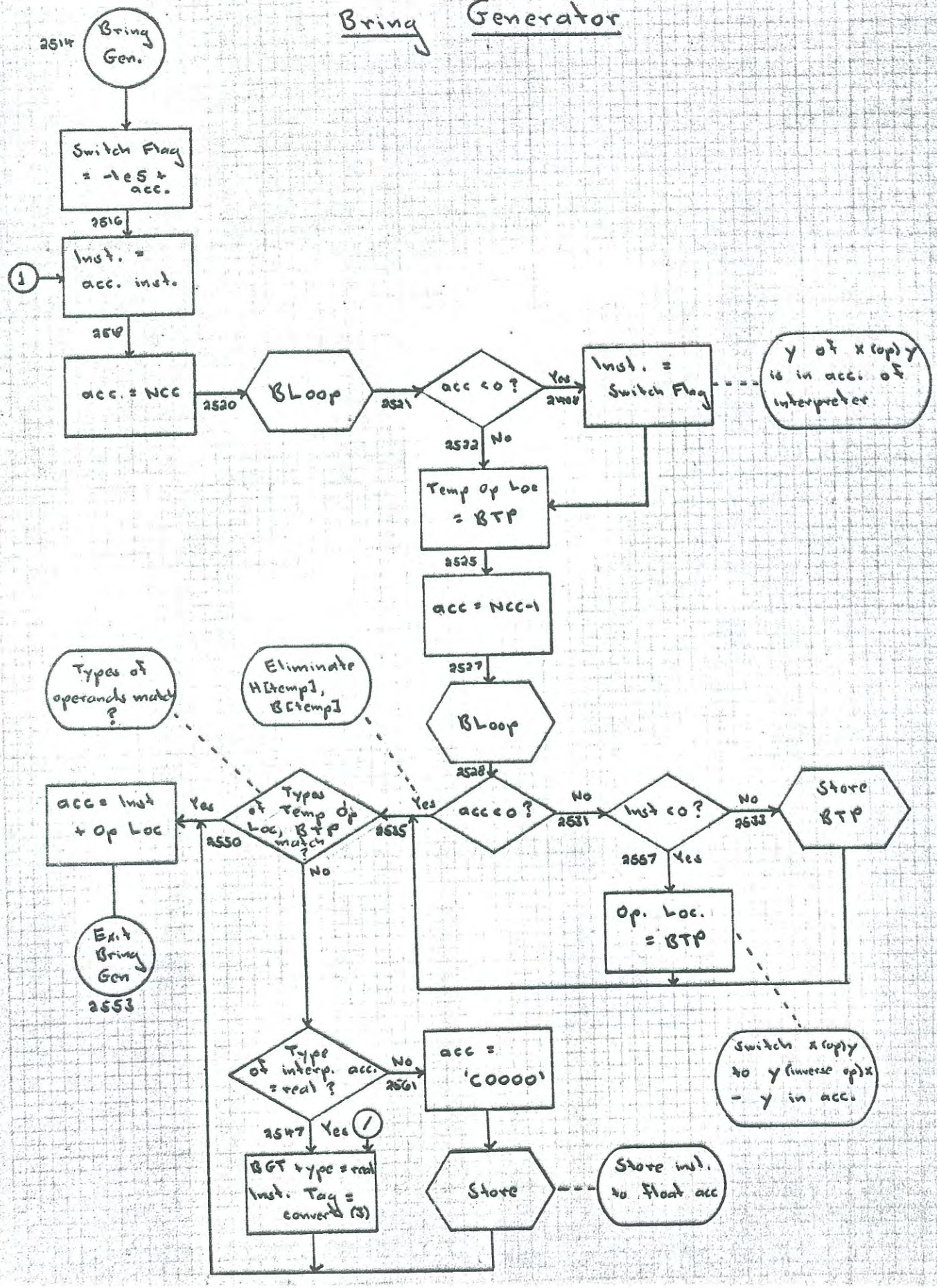

K Subscript
2835
1st Subscript ?
Yes
2930
Unary Bring Gen
2932
End Arith.
No
2946
HTP Tag = real ?
Yes
2948
HTP Tag = Intgr
acc = C4000
2950
Value of 1st subscript now in interp. acc.; HTP set to hold value
2761
No
NCC = NCC-1
Store
2936
<NCC> Type = switch ?
Yes
2940
acc = U4000 + add <NCC> (Ptable loc)
2926
Store
4450
Set HTP negative
2841
No
add. of Tsub = add2 of SCA
2918
SCC = SCC+1
2953
Read
acc = S0000 + Tsub
2930
Store
2136
α = , ?
Yes
2910
NCC = NCC+1
2824
acc = N0000 + Tsub - 1
2827
Store
338
No
Error ***** *****
4428
No
α =] ?
2815
Yes
Set HTP, var. Flag negative
2818
acc = M0000 + add <<NCC>> (array Lo from Ptable)
2821
Store
2825
acc = NCC, sign < 0
2827
Store
This stores a blank for the ZO and places the add. of the blank in <NCC>
1
2828
Erase Flag from E-id in <<CC>, decrease add2 by 2
2832
α-id = +-id
Store and Read

BY_____ DATE_____ SUBJECT_____

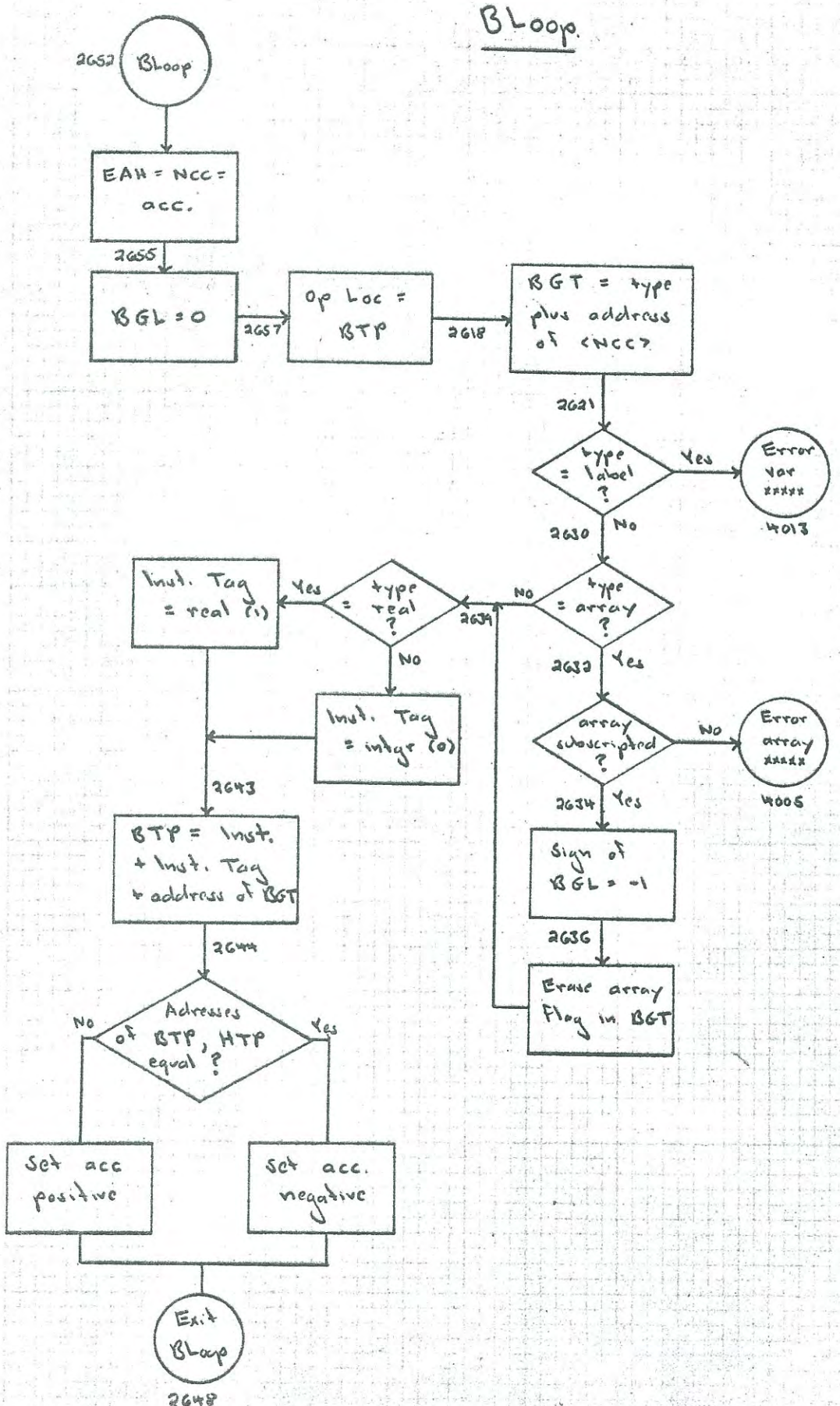
CHKD. BY_____ DATE_____

JOB NO._____

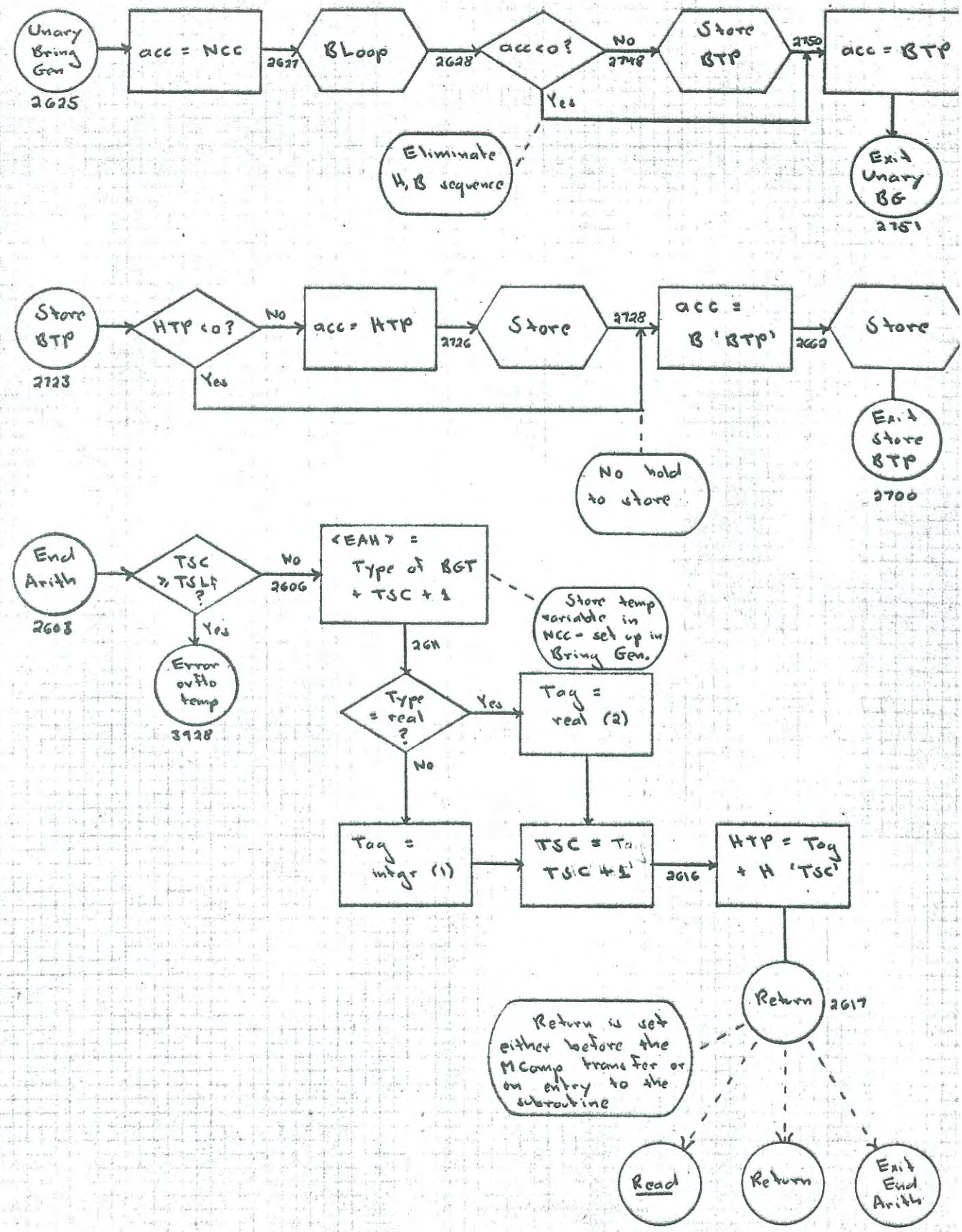
Bring Generator



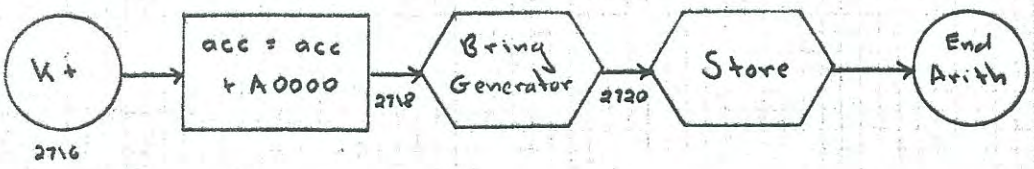
BLoop

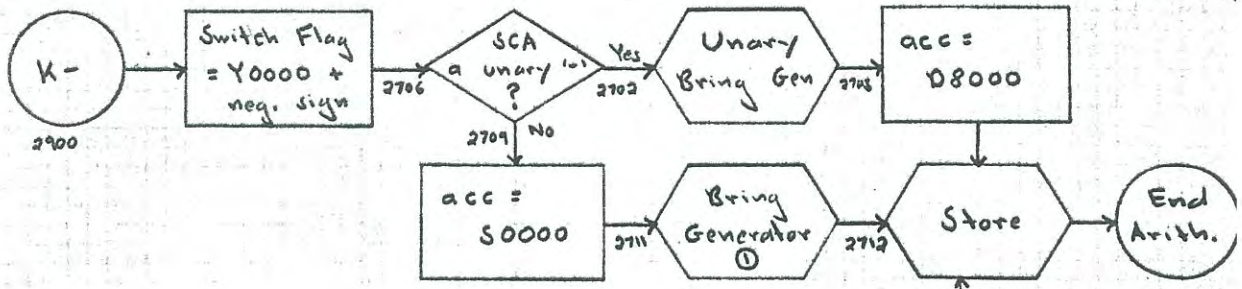

2652
BLoop
EAH = NCC = acc.
2655
BGL = 0
2657
Op Loc = BTP
2618
BGT = type plus address of <NCC>
2621
type = label ?
Yes
Error var *****
4013
No
2630
type = array ?
No
type = real ?
Yes
Inst. Tag = real (1)
No
Inst. Tag = intgr (0)
2634
2633
Yes
array subscripted ?
No
Error array *****
4005
Yes
2634
Sign of BGL = -1
2636
Erase array Flag in BGT
2643
BTP = Inst. + Inst. Tag + address of BGT
2644
Adresses of BTP, HTP equal ?
No
Yes
Set acc positive
Set acc. negative
Exit BLoop
2648

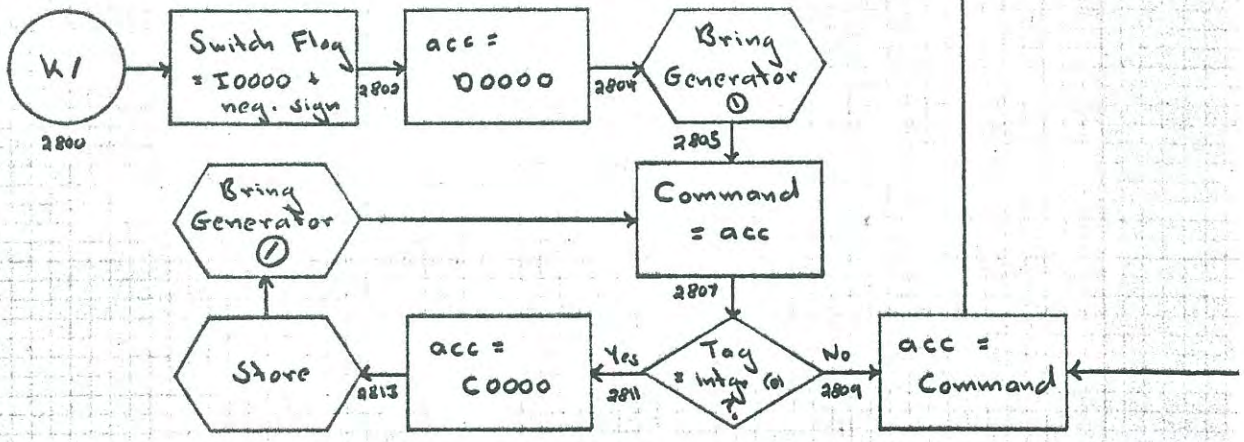
Unary Bring Generator, Store BTP, End Arithmetic

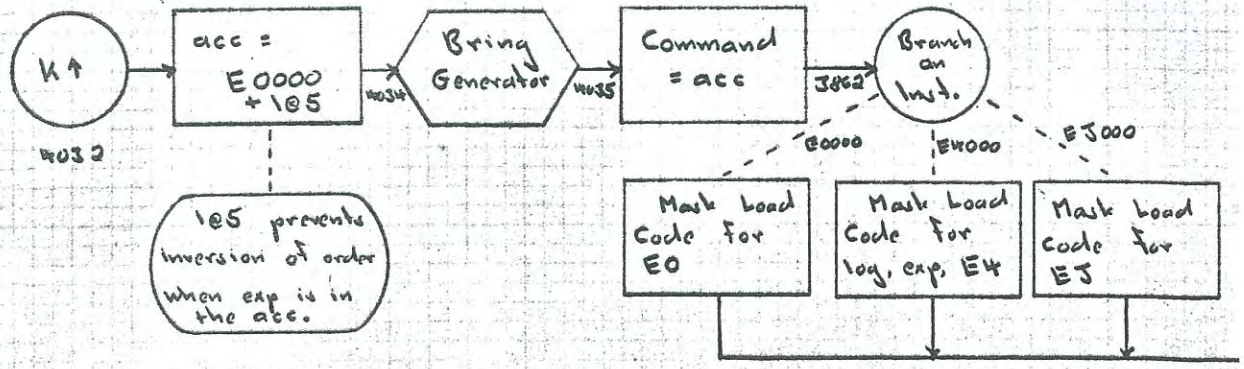


Arithmetic Operations

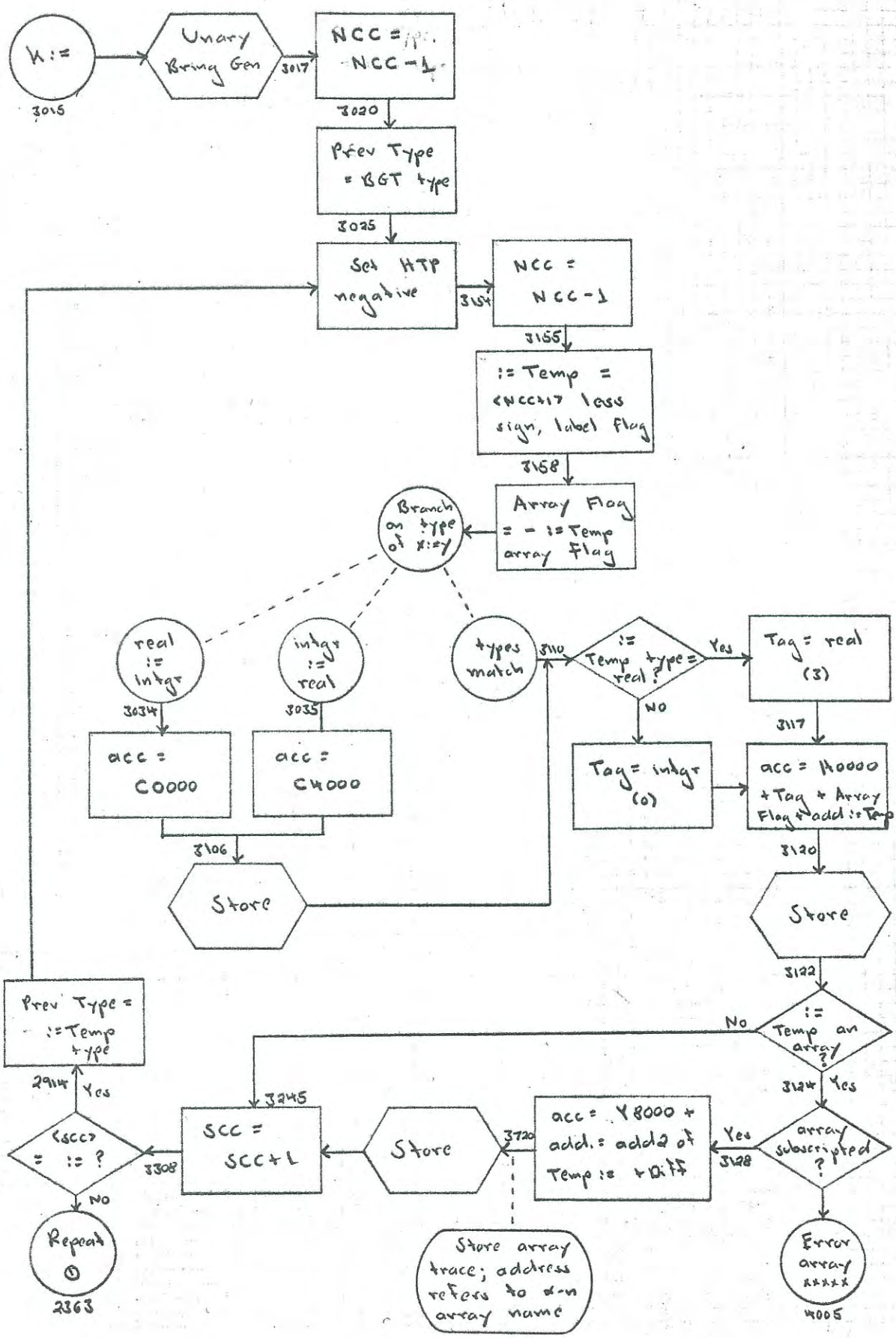




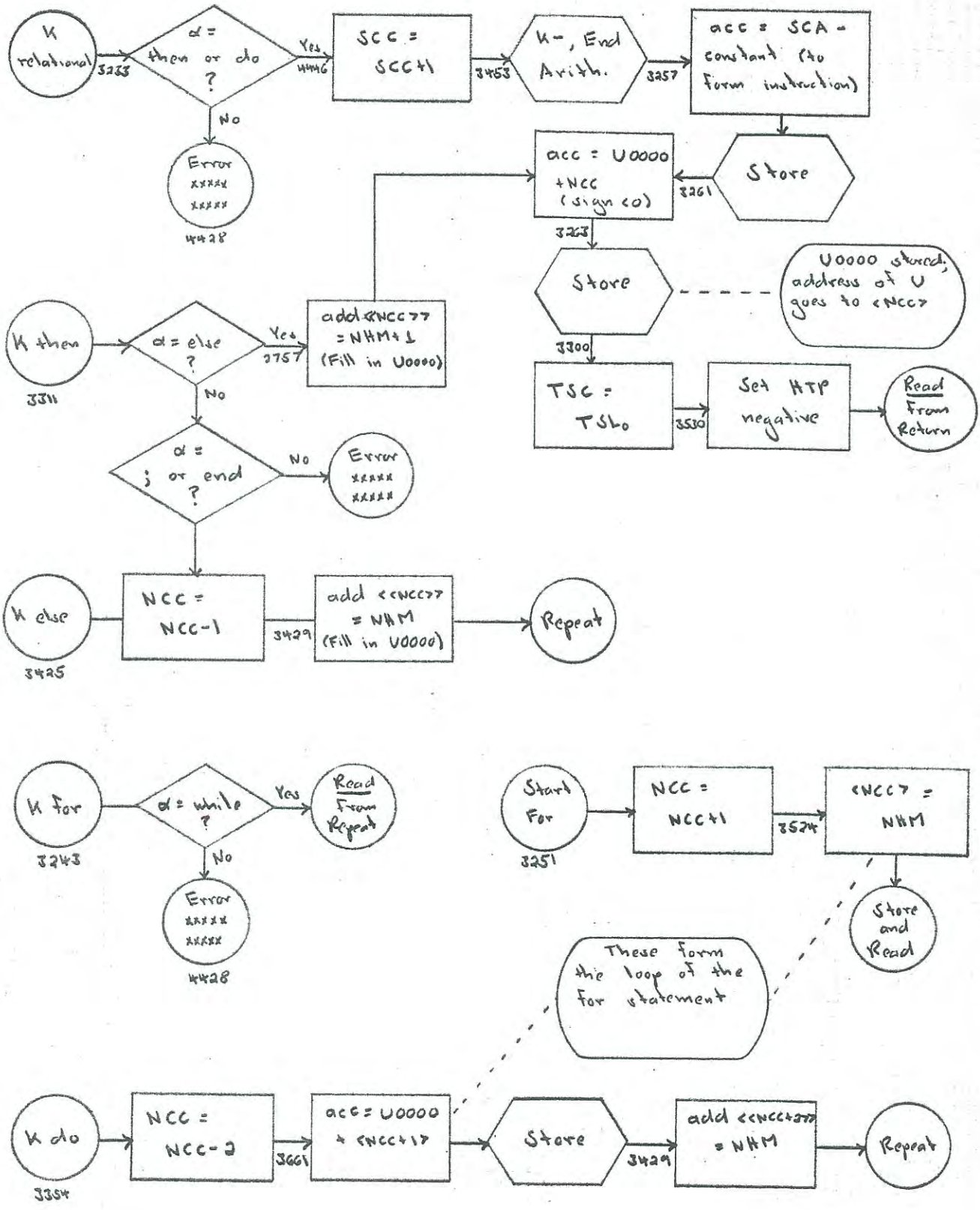




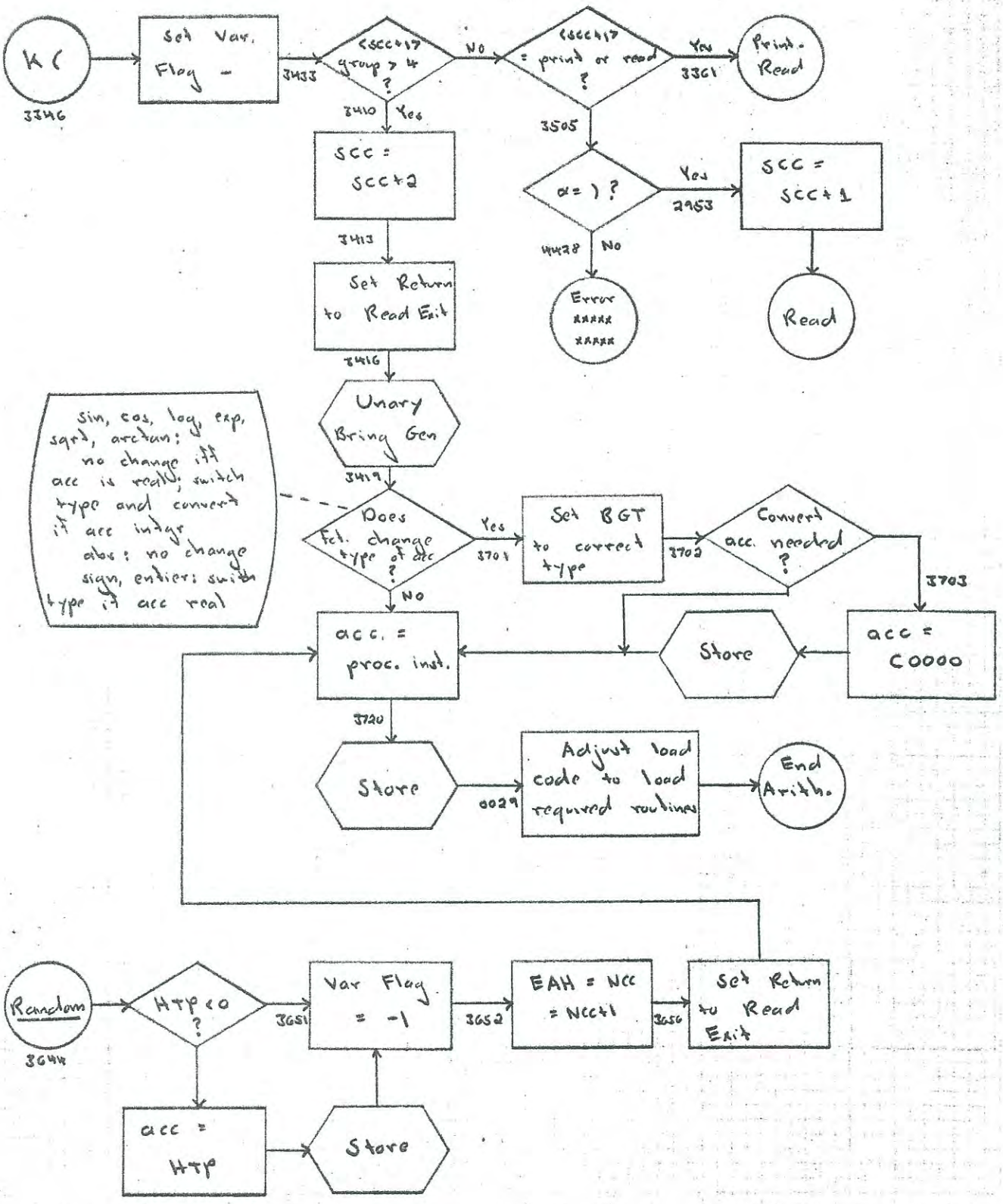
Assignments



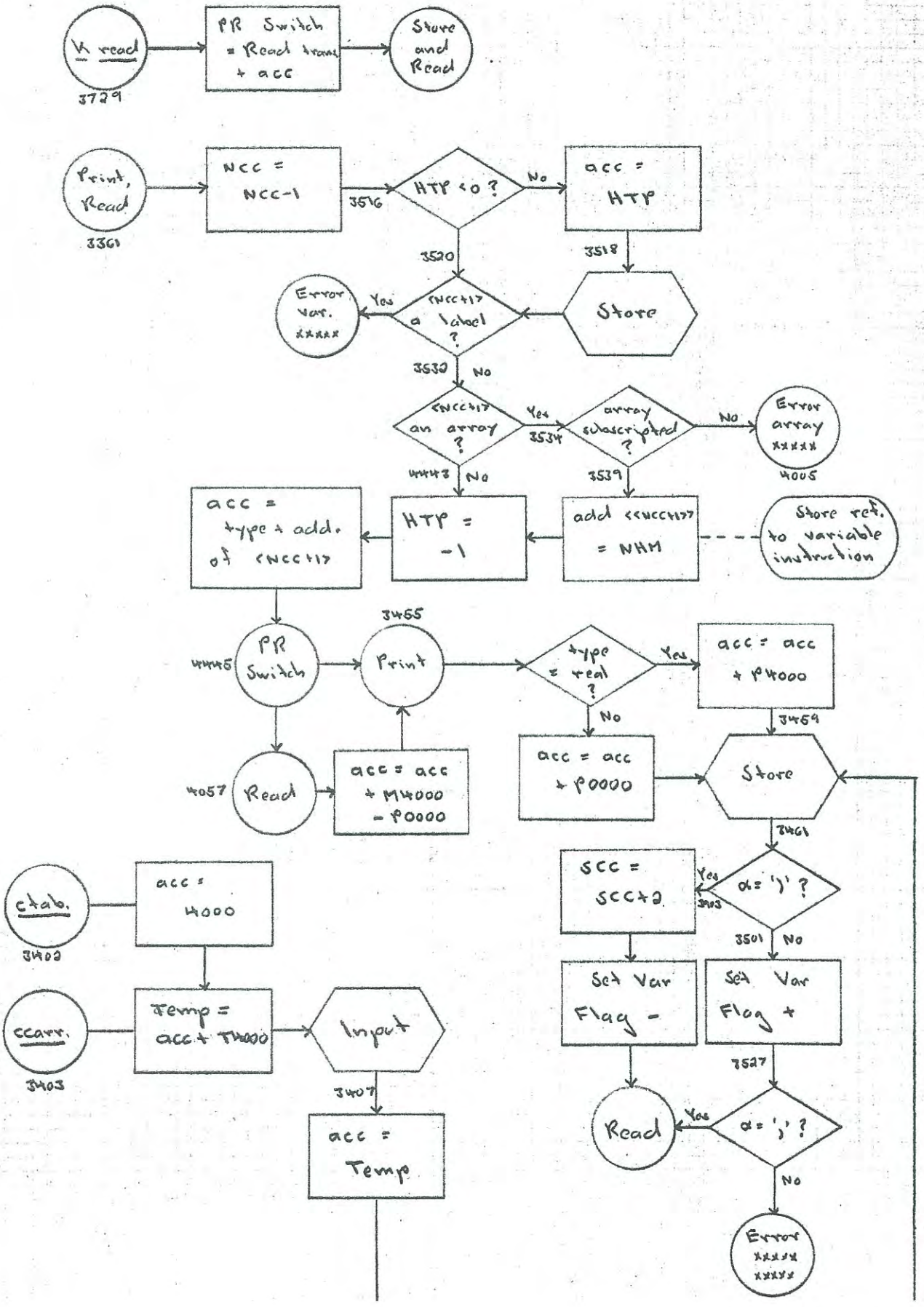
Conditionals, Relationals, For Statements



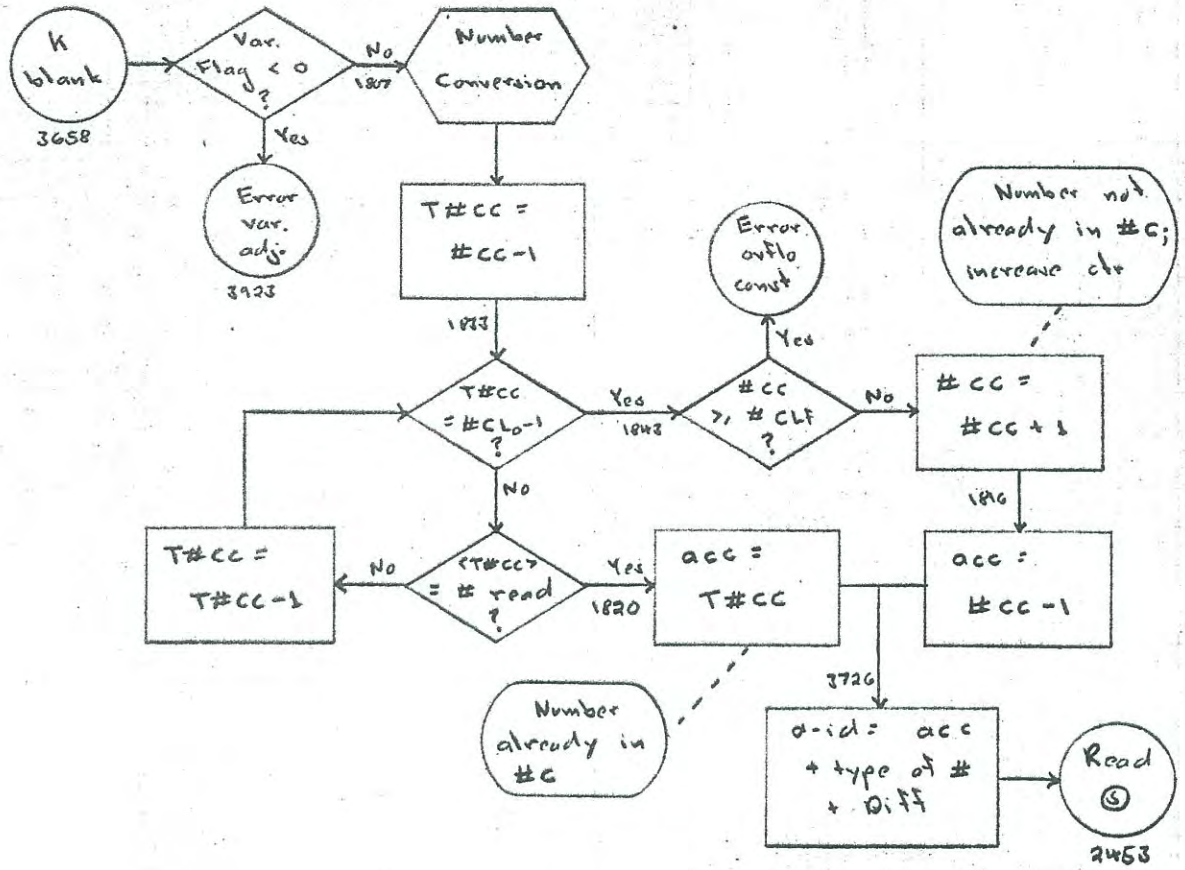
KC , Procedures

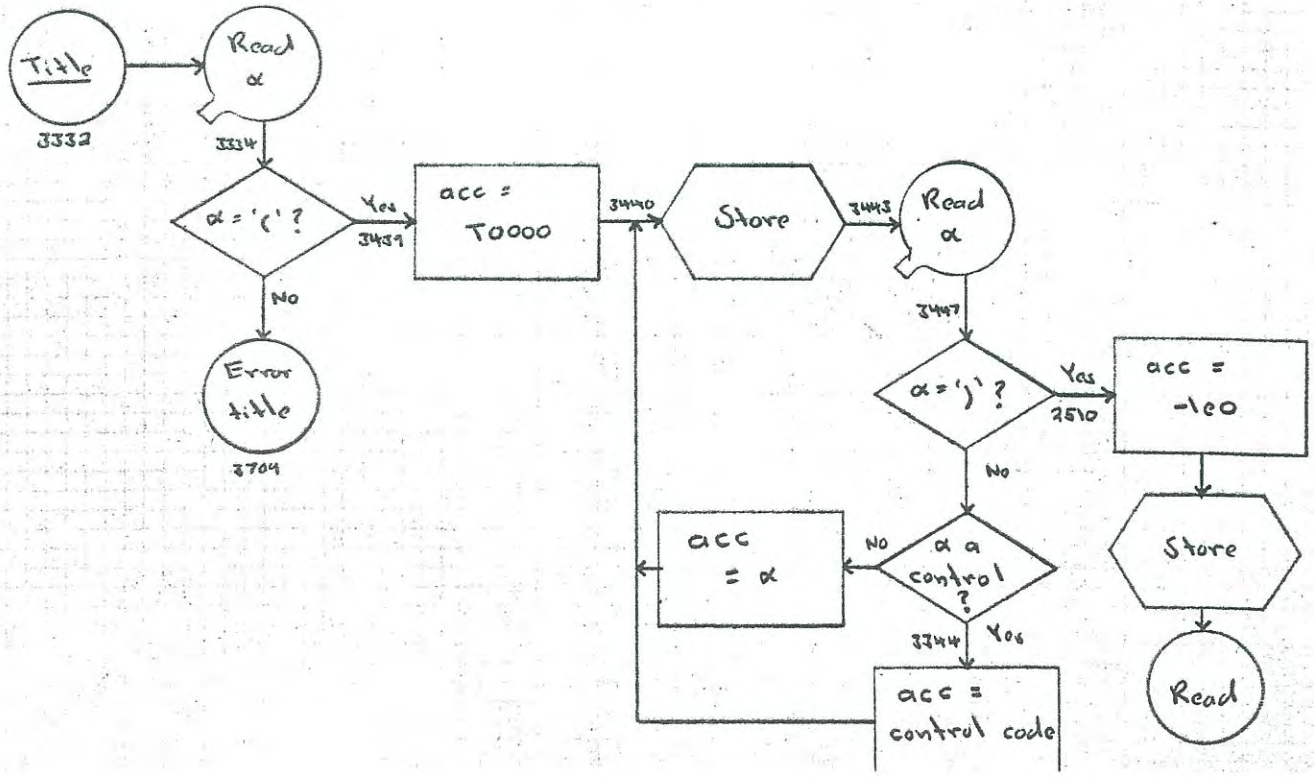


Input - Output

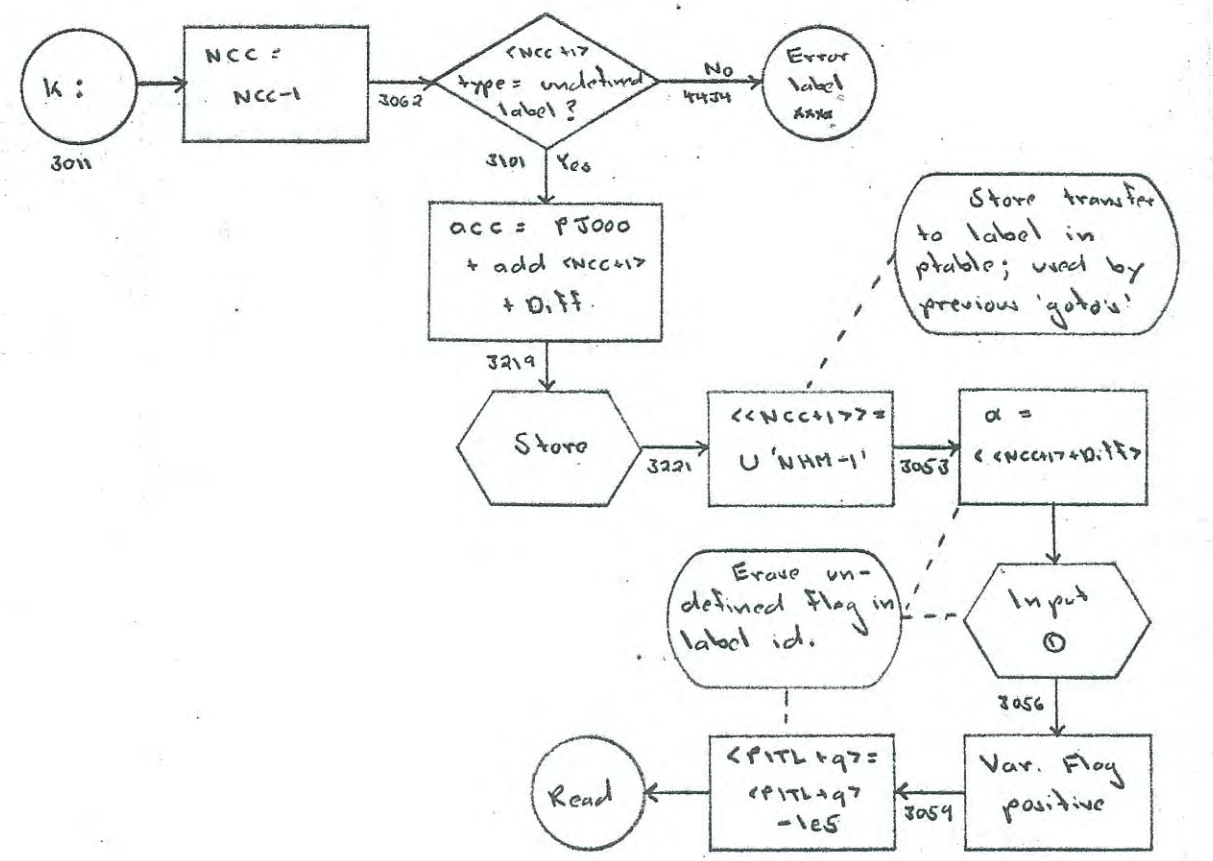


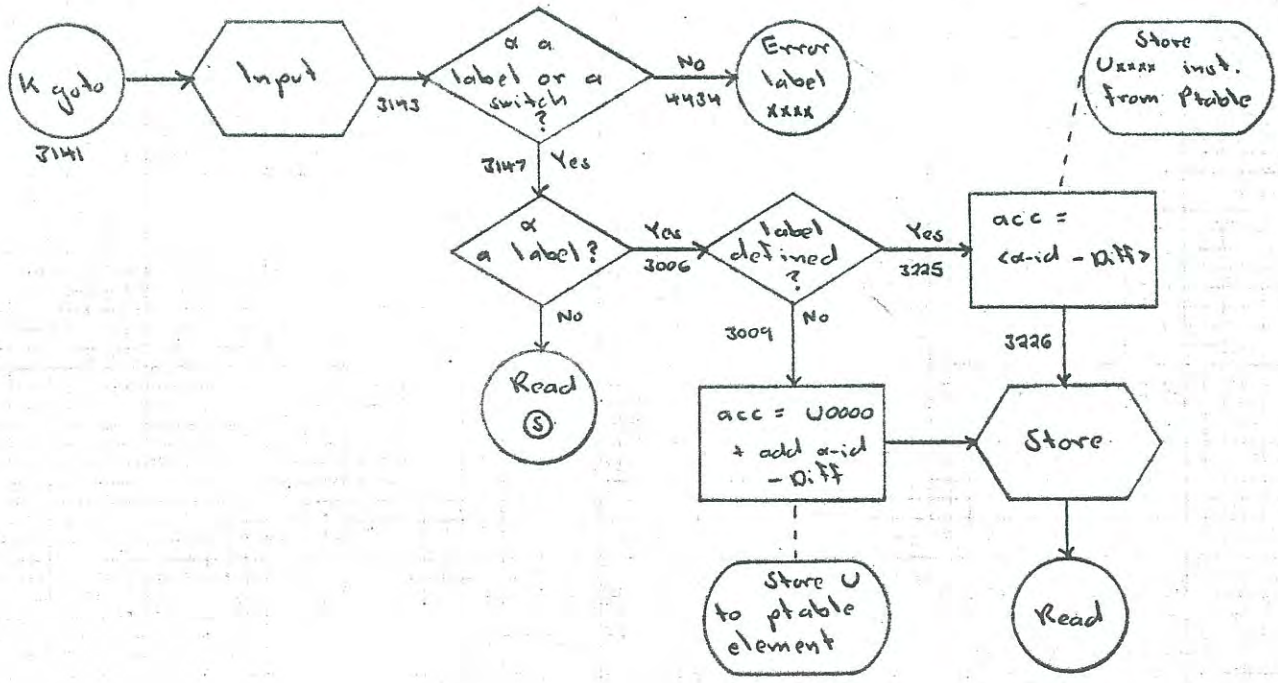
Constants, Title



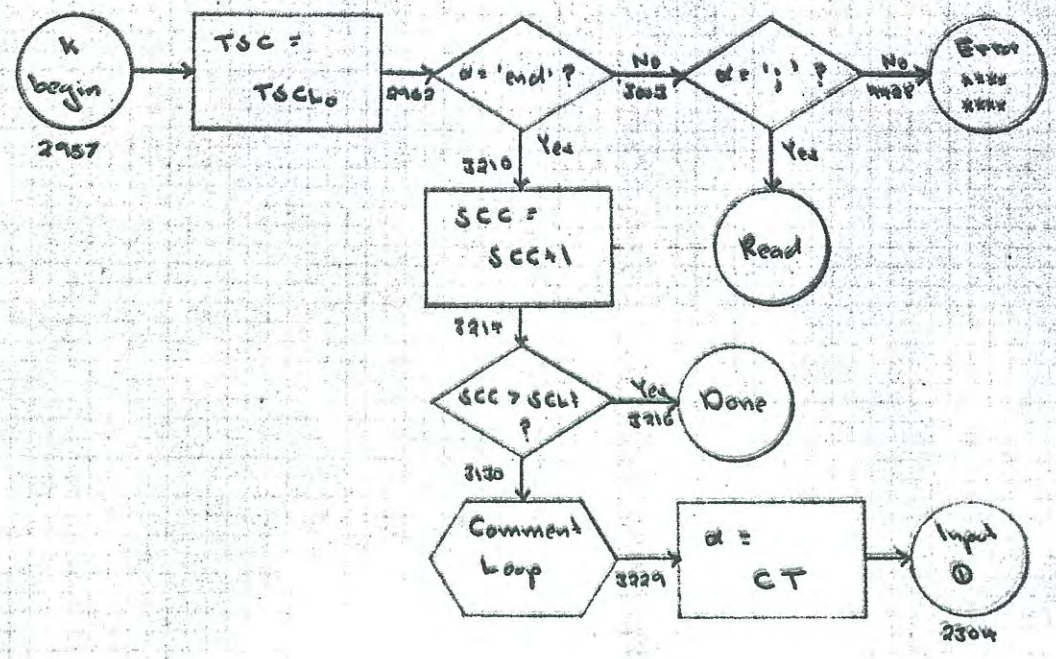


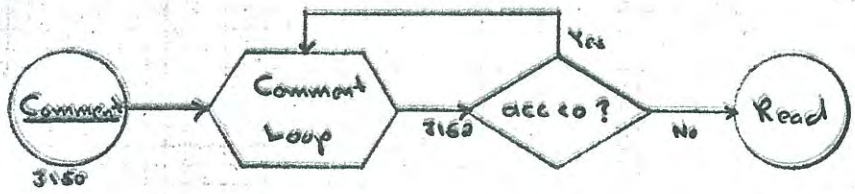
Labels

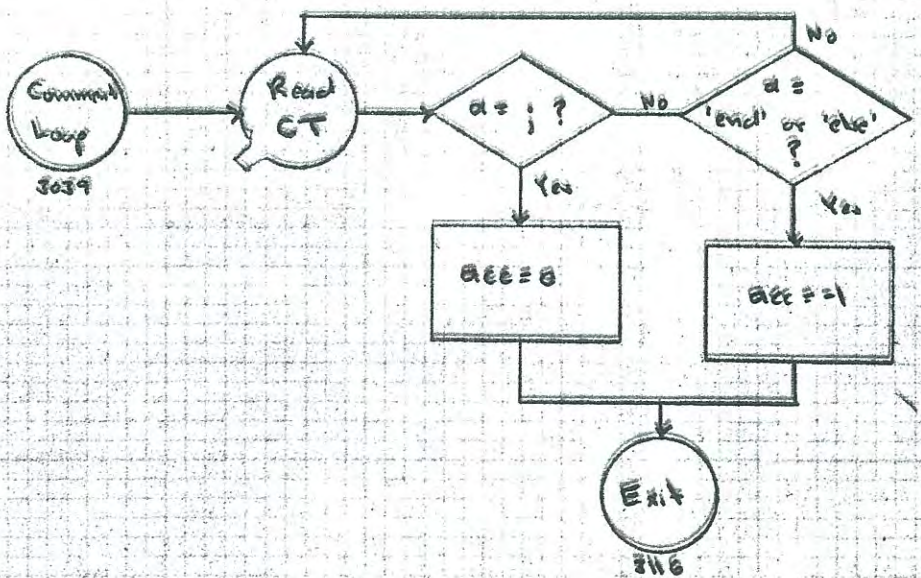

K :
3011
NCC = NCC-1
3062
<NCC+1> type = undefined label ?
No
4434
Error label xxxx
3101
Yes
acc = PJ000 + add <NCC+1> + Diff
3219
Store
3221
<<NCC+1>> = U 'NHM-1'
3053
α = <<NCC+1>+Diff>
Store transfer to label in ptable; used by previous 'goto's'
Input ①
3056
Var. Flag positive
3054
<PITL+q> = <PITL+q> -1e5
Read
Erase undefined flag in label id.


K goto
3141
Input
3143
α a label or a switch ?
No
4434
Error label xxxx
3147
Yes
α a label ?
Yes
3006
label defined ?
Yes
3225
acc = <α-id - Diff>
No
3009
No
Read ⑤
acc = U0000 + add α-id - Diff
Store
3226
Store U to ptable element
Read
Store Uxxxx inst. from Ptable

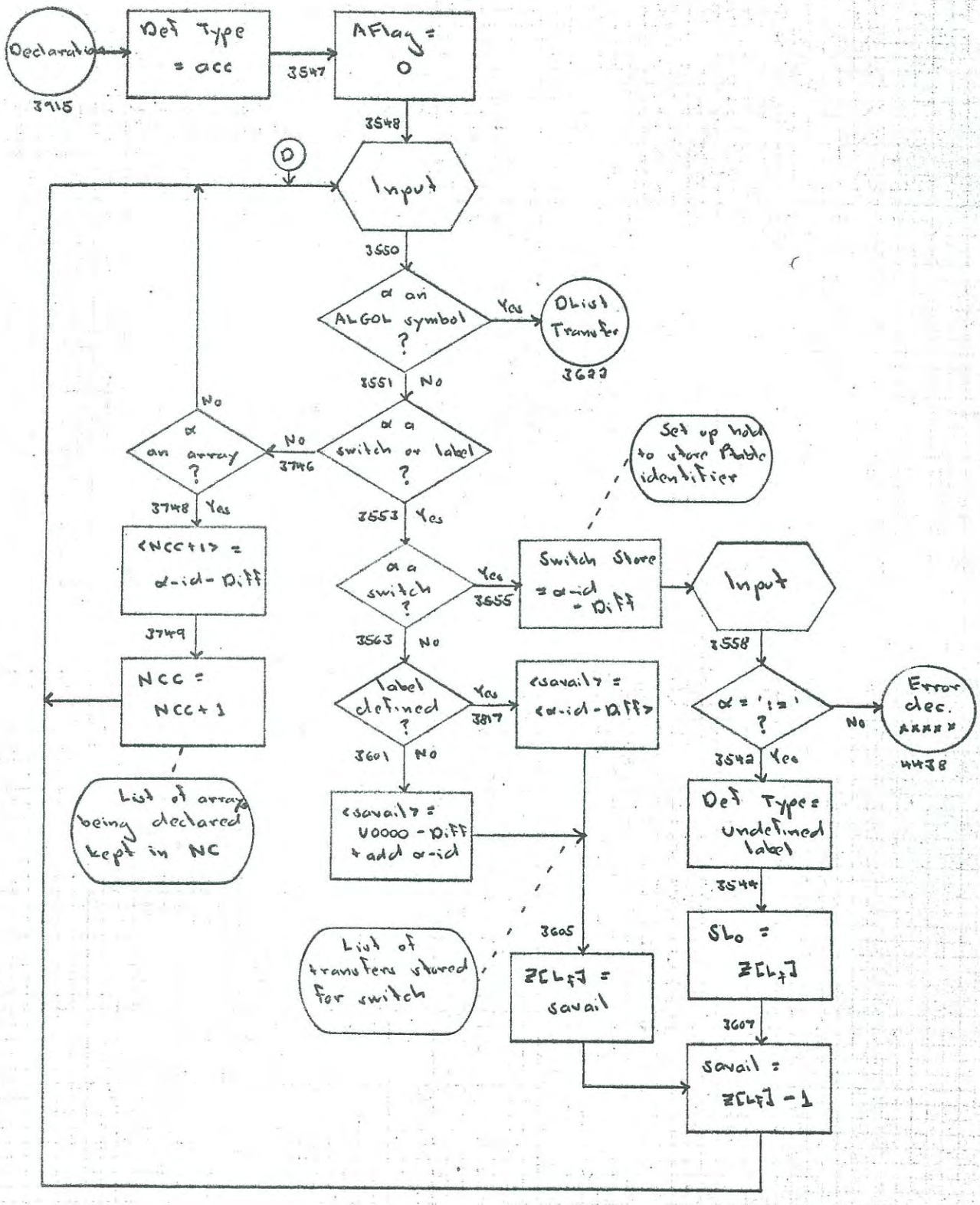
K begin , Comment

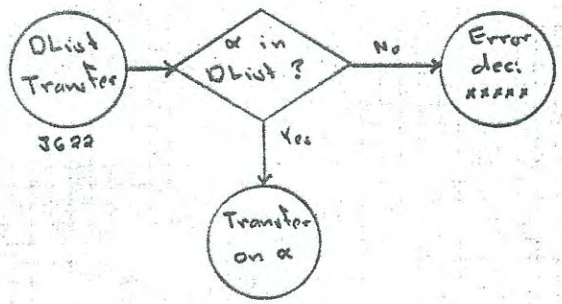


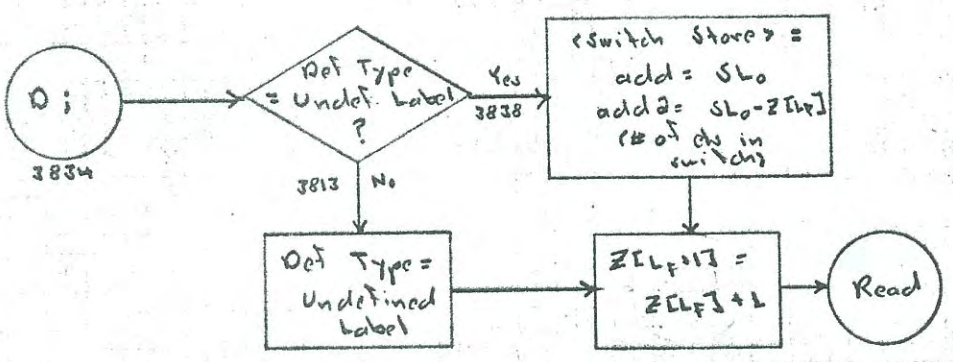


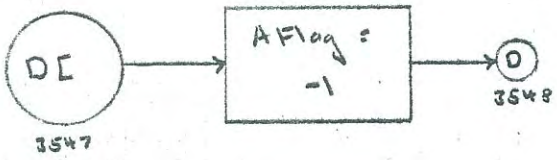


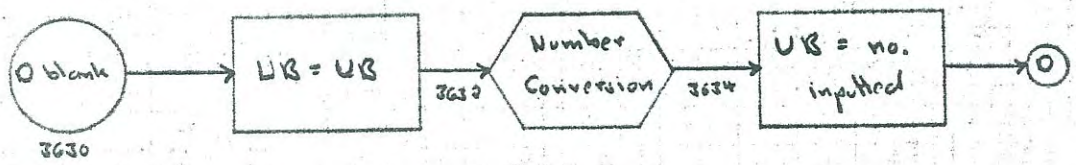
Declarations

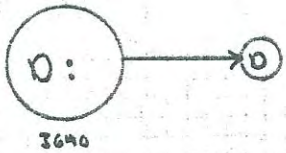


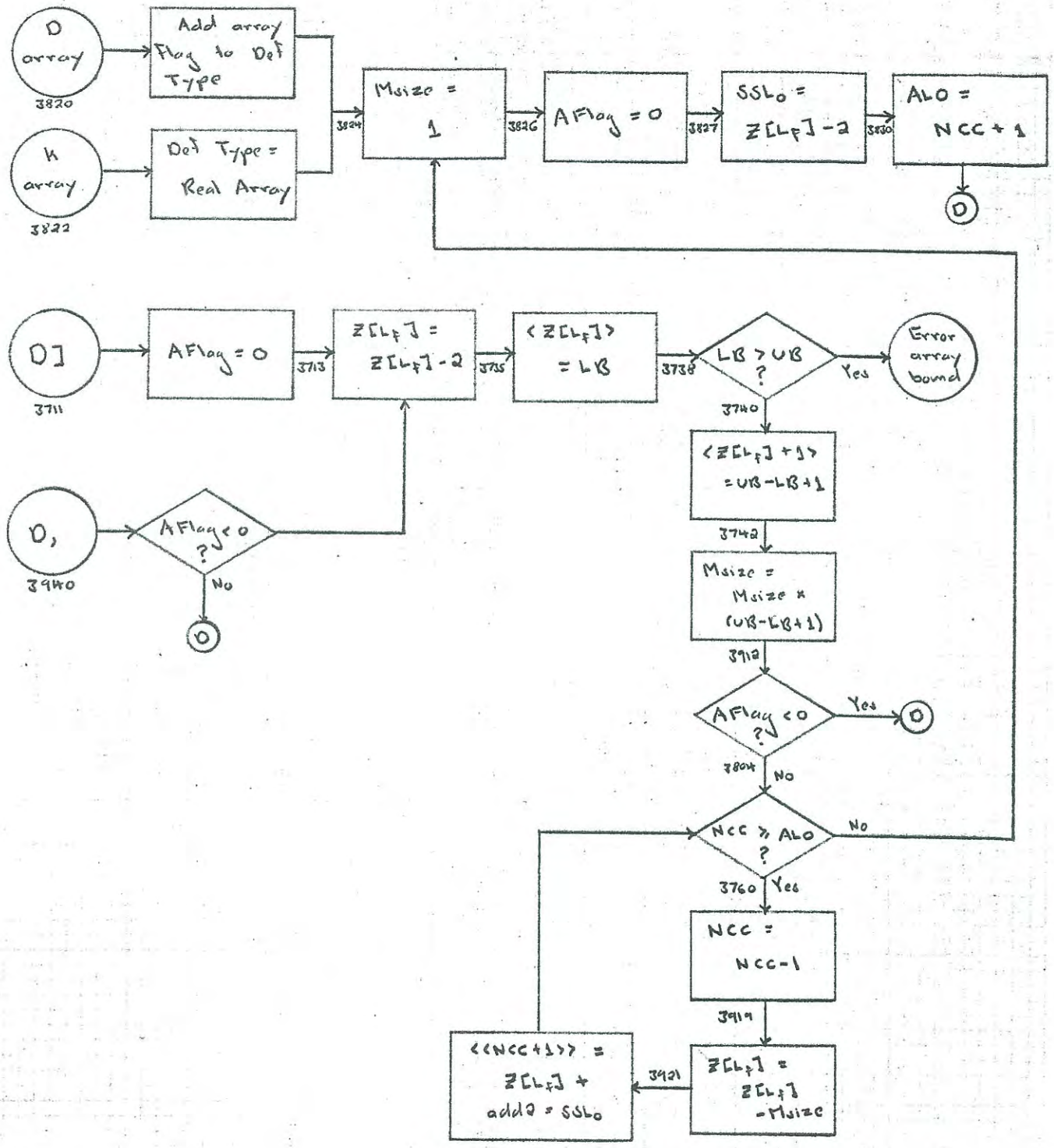

DList Transfer
3622
α in DList ?
No
Error decl: *****
Yes
Transfer on α


D ;
3834
Def Type = Undef. Label ?
Yes
3838
3813
No
<Switch Store> =
add = SL0
add2 = SL0-Z[Lp]
r# of cls in switch
Def Type = Undefined Label
Z[LF+1] = Z[LF]+1
Read

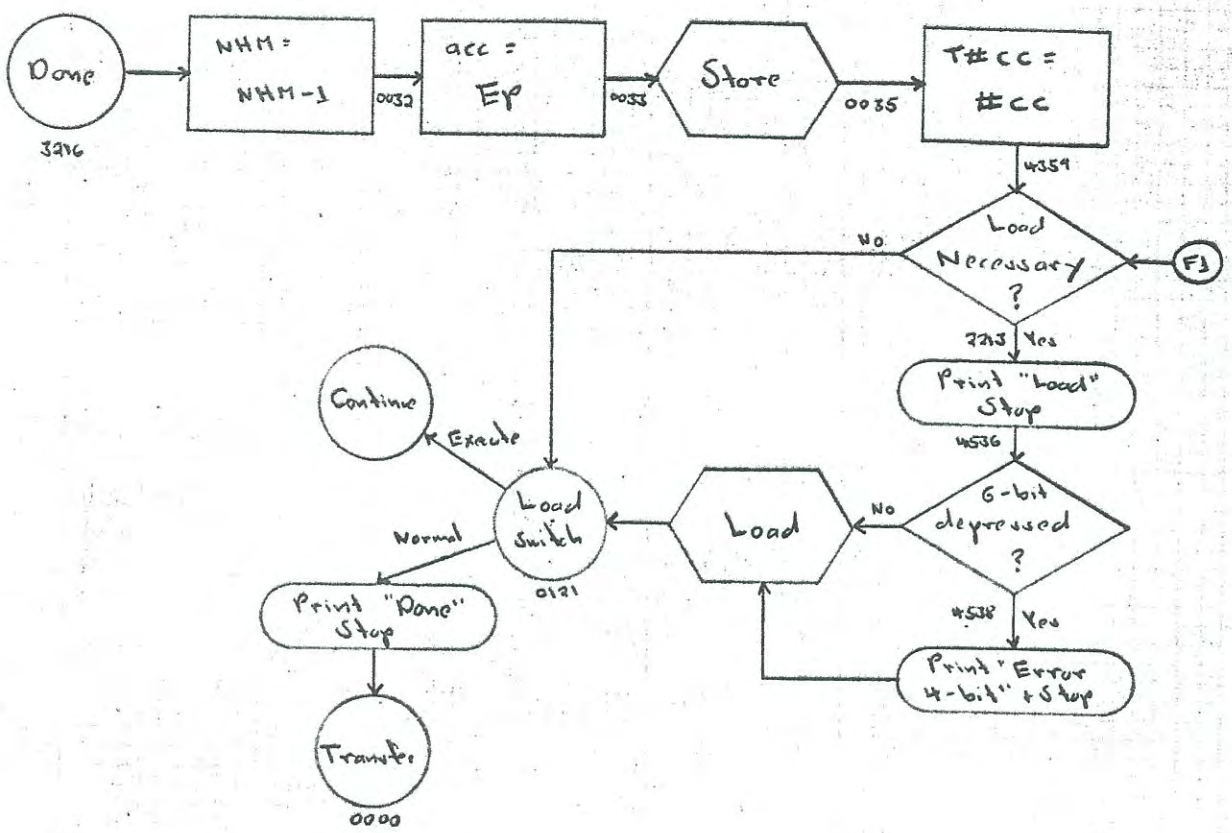

DE
3547
AFlag = -1
D
3548

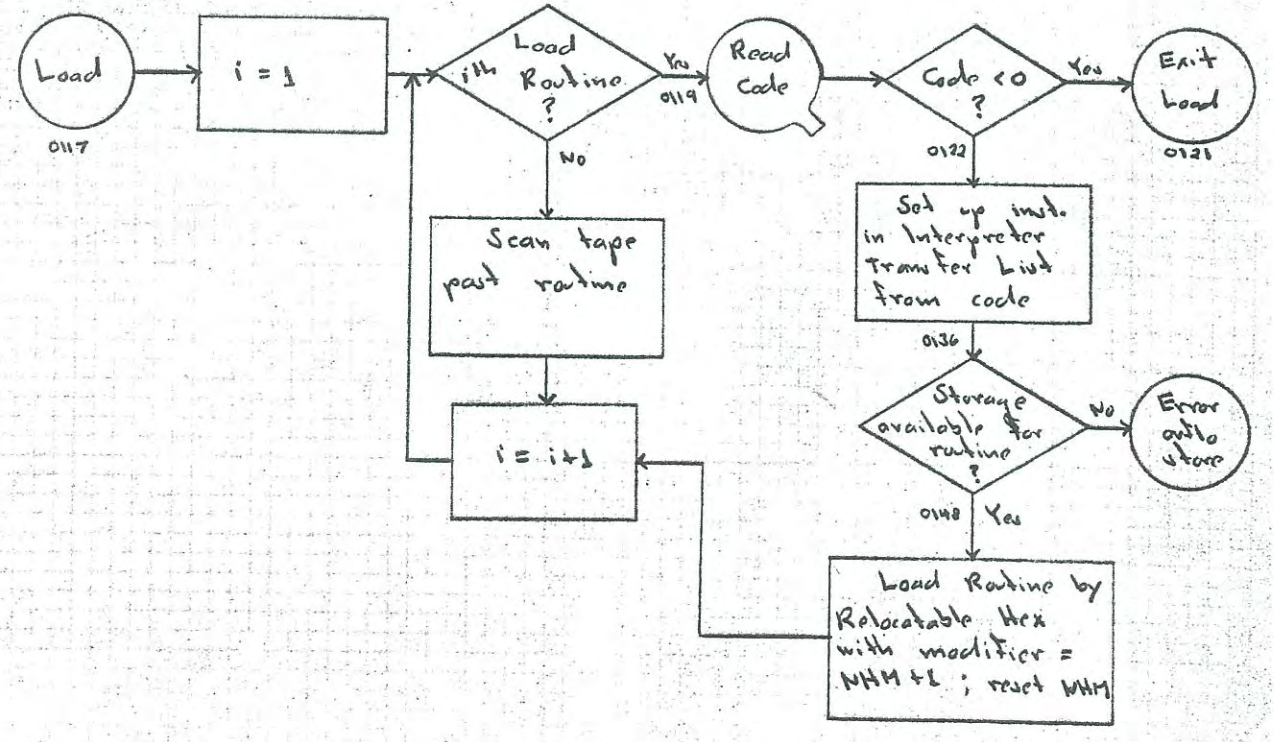

D blank
3630
UB = UB
3633
Number Conversion
3634
UB = no. inputted
D


D:
3640
D

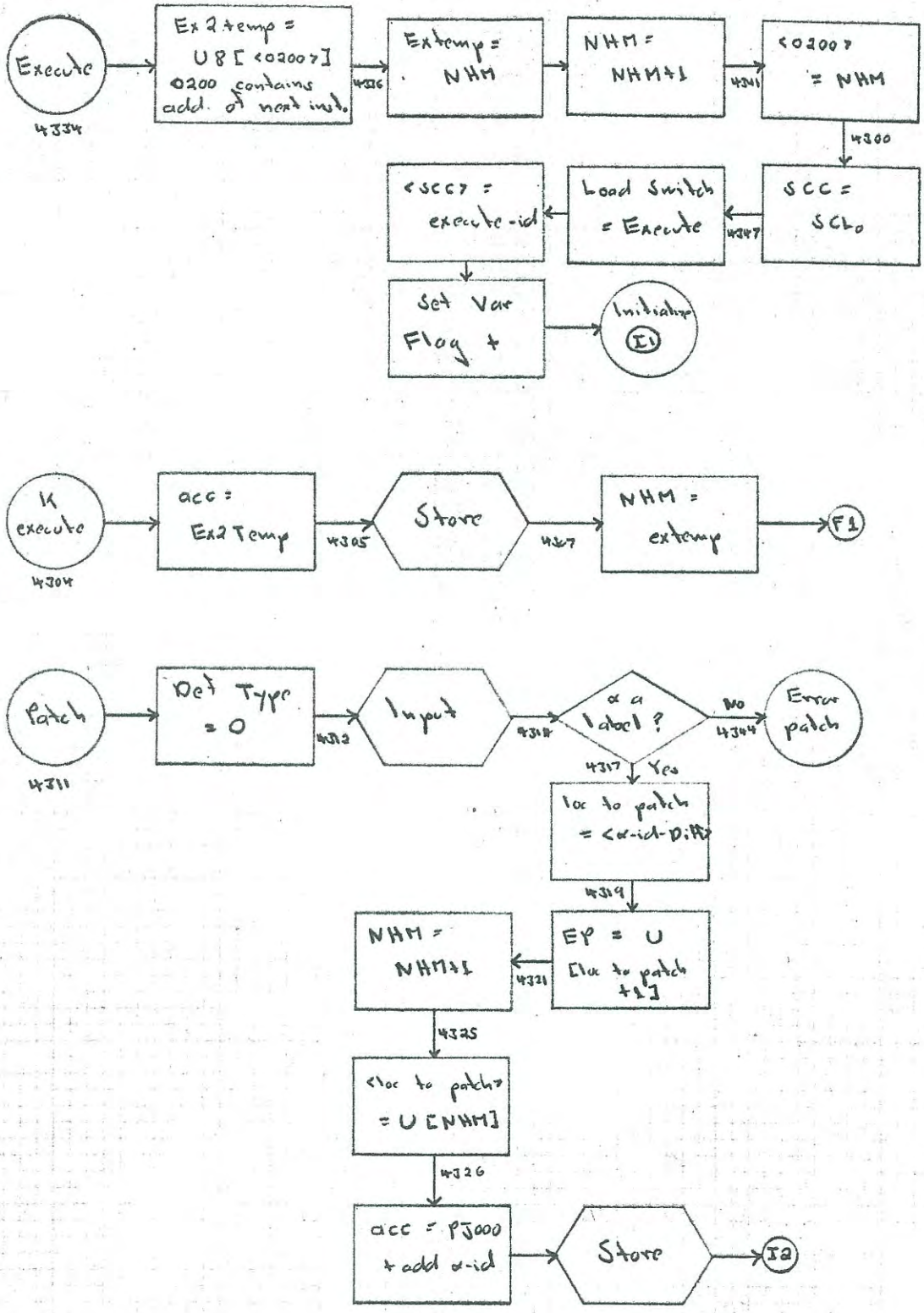

D array
3820
Add array Flag to Def Type
K array
3822
Def Type = Real Array
3824
Msize = 1
3826
AFlag = 0
3827
SSL0 = Z[LF] - 2
3830
ALO = NCC + 1
D
D]
3711
AFlag = 0
3713
Z[LF] = Z[LF] - 2
3715
<Z[LF]> = LB
3738
LB > UB ?
Yes
Error array bound
3740
<Z[LF] + 1> = UB - LB + 1
3742
Msize = Msize * (UB - LB + 1)
3912
D,
3940
AFlag < 0 ?
No
AFlag < 0 ?
Yes
3804
No
NCC > ALO ?
No
3760
Yes
NCC = NCC - 1
3919
<<NCC+1>> = Z[LF] + add2 = SSL0
3921
Z[LF] = Z[LF] - Msize

Done, Load


Done
3216
NHM = NHM-1
0032
acc = Ep
0033
Store
0035
T#cc = #cc
4359
Load Necessary ?
No
F1
7213
Yes
Print "Load" Stop
4536
6-bit depressed ?
No
Load
Load Switch
0121
Execute
Continue
Normal
Print "Done" Stop
Transfer
0000
4538
Yes
Print "Error 4-bit" + Stop

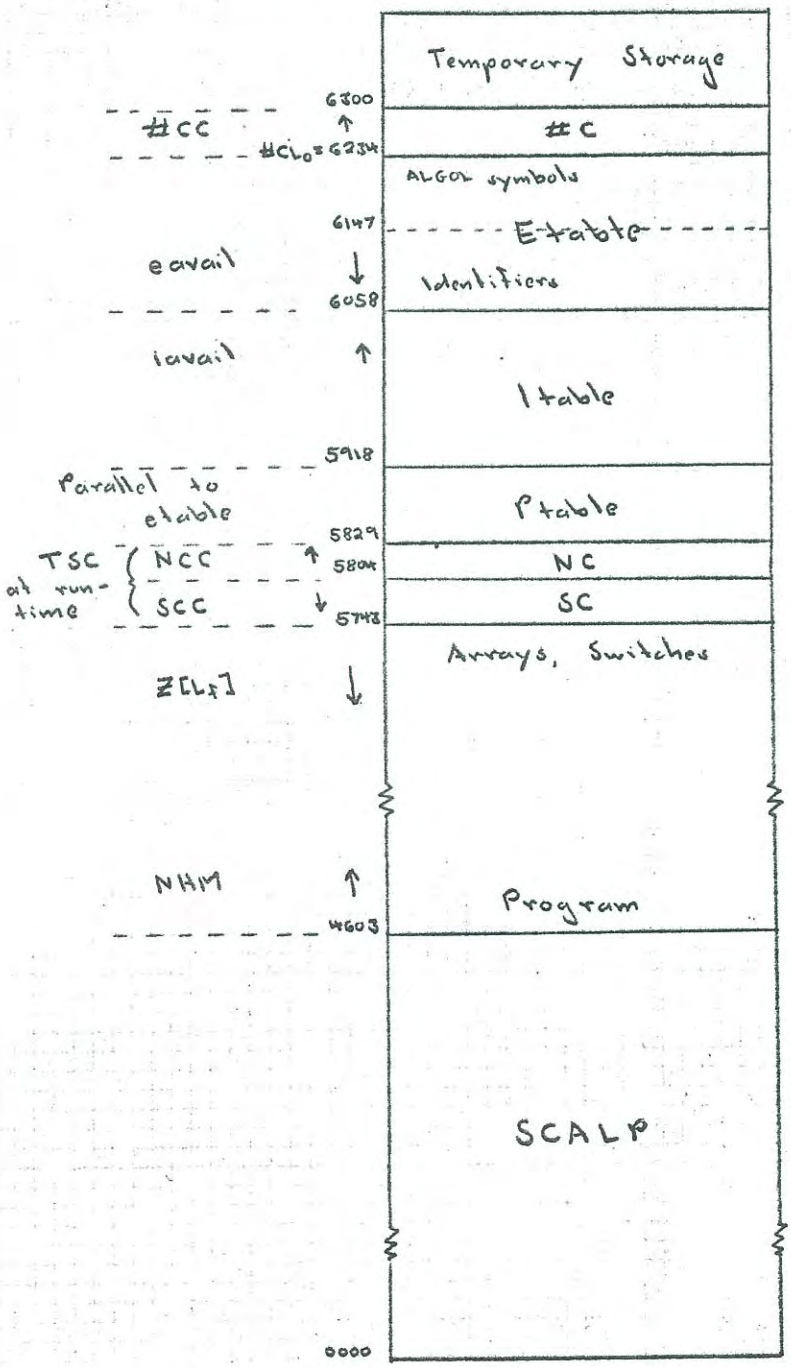

Load
0117
i = 1
Load ith Routine ?
Yes
0119
No
Read Code
Code < 0 ?
Yes
Exit Load
0121
0122
Scan tape past routine
Set up inst. in Interpreter Transfer List from code
0136
i = i+1
Storage available for routine ?
No
Error exits store
0148
Yes
Load Routine by Relocatable Hex with modifier = NHM + L ; reset NHM

Patch, Execute

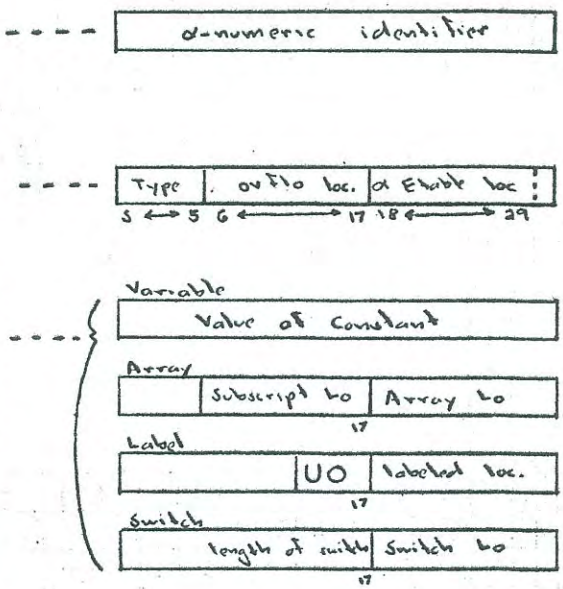


Memory Allocation

Indexing



Word Structure



address (addr) =
bits 18 - 29

add 2 = ovflo =
bits 6 - 17

Type = bits S, 1-5

Instruction (Inst) =
bits 12-17

Tag = bits 16-17 of Inst

Symbols

A: acc -- accumulator; a temporary storage location; used as input to Store

a-id -- identifier of last symbol read

AFlag -- flag which indicates whether Declaration is operating inside of bound pair list

ALo -- location of 1st array being declared in number cellar list

B: BGL -- Bring Generator flag to indicate an array

BGT -- Bring Generator Temp.; contains type of result

BTP -- "Bring To Program" formed by Bring Generator

C: CT -- Comment Temporary; contains a-numeric

D: D -- declare; e.g.; D; means declare;

Def Type - defining type; set at "undefined label" except in Declarations

Diff -- difference between a Etable and Ptable location

E: EAH -- End Arithmetic hold; usually set to store identifier for temporary variable in number cellar

EP -- End Program; contains exit instruction for program

Extemp -- contains reset for NHM after an "Execute"

Ex2temp - contains exit from an "Execute"

H: HTP -- "Hold To Program" ; matched against next BTP to see if both can be eliminated

I: IFlag -- used by interpreter on input to check type

K: K -- "compile" ; e.g., K+ means compile +

L: LB -- lower bound of bound pair in array dec.

Load Code - code word telling which routines on library tape are called for by program

loc to patch - location of PJ inst. to be replaced by a UO to the patch

M: Msize -- size of array being declared

N: NHM -- "next hold to memory" counter

O: Op Loc -- operand location in Bring Generator; i.e. location of y in x+y, etc.

P: PITL -- "primary ITable location" counter for itable reset

Prev -- flag used to determine whether +, - is unary

S: savail -- counter for storing UO instructions in switch declarations

SCA -- identifier of last entry in the symbol cellar

SSLo -- location of 1st subscript constant; used in array declarations

T: T#CC - a) temporary # C counter
b) used to record last value of #CC to use for initializing in "Execute"

TSC -- temporary storage counter

U: UB -- upper bound of bound pair in array declaration

V: Var. Flag - variable flag; tells whether last symbol read was a variable or not

< x > -- contents of location x; e.g., <NCC> = contents of the location whose address is given by NCC

<< x >> -- iteration of < > : contents of the location whose address is in location x

Symbol Types

ALGOL Symbols:

Group 1: Tag = -1@3
then, while, do, else, end, ;,), ',]

Group 2: Tag = -2@3 + n@5
n=0 ↑
n=1 ×, /
n=2 +, - (+ Flag bit @30)
n=3 ≤, ≠, <, >, =, ≥

Group 3: Tag = -3@3
:, blank, array, real, switch, integer, comment, title, goto, go to, c tab., ccarr., /=, random, for, [
Tag = -3@3 + 1@4 : print, read

Group 4: Tag = -4@3 + 1@5
begin, if, :=, (

Group 5: Tag = -5@3
sin, cos, log, exp, sqrt, arctan, abs, sign, entier

Variables:

1@1 - label
1@2 - not used
1@3 - array (switch with 1@1)
1@4 - array subscript flag (+1 only in NC)
1@5 - real flag; 0 means integer

SCALP

Library Tape

Each routine should have an even multiple of 6 instructions, and is punched by the ALGOL relocatable hex punch routine. Immediately preceeding each routine is a code word telling:

1) the location of the subroutine call (in track 03) @ q=29

2) (n-1) @ q=11, where n = no. locations required

Tape format: 8zzz zzzz'

{code routine} repeated 10 times

80000000'

Order of routines:

E0
ln
E4
exp
EJ
sin-cos
arctan
sqrt
random
entier