LGP-30

ROYAL PRECISION ELECTRONIC DIGITAL COMPUTER

ACT I COMPILER

ROYAL McBEE CORPORATION • *data processing division*

PREFACE

THE ACT I SYSTEM

THE ACT I COMPILER (ALGEBRAIC COMPILER AND TRANSLATOR) IS A NEW PROGRAMMING DEVELOPMENT OF THE ROYAL MCBEE CORPORATION. ITS PRIME PURPOSE IS TO FACILITATE THE CODING OF PROBLEMS THROUGH THE USE OF COMMON MATHEMATICAL TERMINOLOGY, BEARING IN MIND SOME BASIC RULES OF ELEMENTARY ALGEBRA. FIELD TESTED AND PROVEN, THIS NEW TECHNIQUE WILL PROVE USEFUL TO BOTH THE INEXPERIENCED AND EXPERIENCED PROGRAMMER IN TERMS OF MORE RAPID PROBLEM SOLUTION.

ONE INSTRUCTION IN ACT I MAY BE THE EQUIVALENT OF MANY MACHINE LANGUAGE INSTRUCTIONS SIMPLIFYING COMMUNICATION WITH THE COMPUTER. THROUGH THIS SIMPLIFICATION OF PROBLEM EXPRESSION, MANY HUMAN CODING ERRORS CAN ALSO BE OBVIATED.

ACT I HAS TWO DISTINCT PHASES. FIRST, THE TIME SPENT TO TRANSLATE THE COMMON LANGUAGE INTO DETAILED LANGUAGE - (COMPILE TIME) AND SECOND, THE TIME REQUIRED TO EXECUTE THE MACHINE LANGUAGE PROGRAM THAT HAS BEEN GENERATED - (COMPUTE TIME).

SINCE THE FIRST AND SECOND PHASE MAY BE SEPARATE TIMES, THE COMPILER NEED NOT BE IN THE LGP-30 AT COMPUTE TIME. THIS MEANS THE ENTIRE COMPUTER MEMORY IS AVAILABLE FOR USEFUL CALCULATIONS AT COMPUTE TIME.

I WISH TO ACKNOWLEDGE MY APPRECIATION TO THE MANY PEOPLE WHO OFFERED SUGGESTIONS AND CRITICISMS OF THE ACT I SYSTEM. IN PARTICULAR DR. HENRY BOWLDEN OF NATIONAL CARBON COMPANY WHO OFFERED SOME OF THE BASIC CONCEPTS AND MEL KAYE OF ROYAL MCBEE WHO DID THE BULK OF THE PROGRAMMING.

CLAY S. BOSWELL, JR.

TABLE OF CONTENTS

		PAGE
	PREFACE	
I	INTRODUCTION	1
II	DEFINITIONS AND LIMITATIONS OF TERMS	2
	1. SYMBOLS	
	2. OPERATIONS	
	3. CONSTANTS	
	4. BRACKETS	
	5. STATEMENT SYMBOLS	
	6. REGION SYMBOLS	
III	SPECIAL STATEMENT FORMS	5
	DIM, INDEX, ITER, SUB	
IV	EXAMPLES	6
	1. VARIANCE	
	2. PRODUCTION & DISTRIBUTION	
V	OPERATIONS	9
VI	USE	11
	1. PUNCHING PROCEDURE	
	2. COMPILING PROCEDURE	
	3. PUNCH OUT PROCEDURE	
	4. PROCEDURE TO EXECUTE PROGRAM	
	5. PROCEDURE TO CHECKOUT PROGRAM	
	6. ERROR CODES	
	7. PROGRAMMING LIMITATIONS	
VII	APPENDIX	14
	1. SCALING TECHNIQUE	
	2. BRACKETS AND PRECEDENCE	
	3. SUBROUTINES AND CALLING SEQUENCE	
	4. ASSEMBLY FOR ACT I	

I INTRODUCTION

THE ROYAL MCBEE ACT I PROGRAMMING SYSTEM WAS DESIGNED TO MAKE THE PROGRAMMER'S JOB OF CODING AS PAINLESS AS POSSIBLE. PROGRAMMING INFORMATION IS PRESENTED TO THE LGP-30 IN MUCH THE SAME MANNER AS MIGHT BE USED TO EXPLAIN A PROBLEM TO A COLLEAGUE. THE COMMON BASIS FOR THE EXCHANGE OF MATHEMATICAL IDEAS IS THE LANGUAGE OF ALGEBRA. THIS COMMON LANGUAGE OF ALGEBRA IS THE BASIC LANGUAGE OF THE ACT I PROGRAMMING SYSTEM. IT HAS BEEN AUGMENTED SOMEWHAT BEYOND BASIC ALGEBRA TO GIVE THE PROGRAMMER THE POWER TO INSERT LOGICAL DECISIONS INTO HIS PROGRAM AND TO WORK WITH ARRAYS OR SETS OF PARAMETERS AS READILY AS HANDLING ONE VARIABLE OR CONSTANT. THIS NEW LANGUAGE IS KNOWN AS THE "SOURCE" LANGUAGE; IT IS THIS LANGUAGE THAT THIS MANUAL DEALS WITH.

HOW THE COMPILER WORKS IS BASICALLY HOW ELEMENTARY ALGEBRA WAS LEARNED. WHEN THE FRESHMAN WAS GIVEN THE STATEMENT THAT FOLLOWS, HE LEARNED TO WORK FROM THE INNERMOST BRACKETS OUT AND TO PERFORM FUNCTIONS (SUCH AS SIN, COS, LOG) BEFORE MULTIPLICATIONS AND DIVISIONS WHICH, IN TURN, MUST BE DONE BEFORE ADDITIONS AND SUBTRACTIONS.

$$c \times \sin (a + b) + d$$

IN ORDER FOR THIS STATEMENT TO MEAN THE SAME THING TO EVERYONE AND TO THE LGP-30, THE OPERATIONS MUST BE EXECUTED IN THE SAME SEQUENCE. THAT IS:

FIRST $a \underline{+} b$	BECAUSE IT IS THE INNERMOST BRACKET LEVEL
SECOND $\underline{\sin} (a + b)$	BECAUSE IT IS A FUNCTION
THIRD $c \underline{\times} \sin (a + b)$	BECAUSE IT IS A MULTIPLICATION
FOURTH $c \times \sin(a+b) \underline{+} d$	BECAUSE IT IS AN ADDITION

THE RESULT OF THE "SOURCE" PROGRAM WILL BE AN EFFICIENT MACHINE LANGUAGE LGP-30 PROGRAM KNOWN AS THE "OBJECT" PROGRAM. IN MANY CASES THIS RESULTING "OBJECT" PROGRAM WILL BE AS SHORT AND FAST AS A HAND PREPARED PROGRAM - IN SOME CASES SHORTER. THE "OBJECT" PROGRAM MAY BE PUNCHED OUT ON TAPE FOR FUTURE USE AND/OR IT MAY BE EXECUTED IMMEDIATELY.

THE PRIMARY INTEREST OF THIS MANUAL IS THE "SOURCE" PROGRAM LANGUAGE: A SERIES OF ALGEBRAIC AND LOGICAL STATEMENTS THAT THE PROGRAMMER WILL USE TO DEFINE HIS PROBLEM. IT SEEMS TO BE A SAFE ASSUMPTION THAT THE PROGRAMMER WILL OCCASIONALLY MAKE AN ERROR IN LOGIC AND/OR ERROR IN TRANSCRIBING HIS PROBLEM INTO "SOURCE" LANGUAGE. SINCE LITTLE WOULD BE GAINED IF THE PROGRAMMER HAD TO CHECK OUT HIS PROGRAM IN MACHINE LANGUAGE, PROVISION HAS BEEN MADE FOR THE PROGRAMMER TO CHECK OUT HIS PROGRAM IN "SOURCE" LANGUAGE. (THE METHOD USED TO ACCOMPLISH THIS WILL BE COVERED IN A LATER SECTION).

IN ADDITION TO THE ALGEBRAIC METHOD OF CODING, BASIC LGP-30 MACHINE LANGUAGE MAY BE USED. THESE OPERATIONS ARE LISTED IN TABLE II (FIGURE 4) OF SECTION V. THE PROGRAMMER MAY USE THE MACHINE LANGUAGE INSTRUCTIONS WHEN A SPECIAL FUNCTION THAT IS NOT REPRESENTED IN TABLE I (FIGURE 3) IS DESIRED. IF ONLY MACHINE LANGUAGE OPERATIONS ARE USED IN CODING A PROGRAM, THE ACT I SYSTEM WOULD THEN BEHAVE IN THE SAME MANNER AS A SYMBOLIC ASSEMBLY PROGRAM. THE INSTRUCTIONS IN TABLE II (FIGURE 4) WILL NOT BE EXPLAINED IN DETAIL IN THIS MANUAL.

II DEFINITIONS AND LIMITATIONS OF TERMS

A SOURCE PROGRAM IS MADE UP OF STATEMENTS. A STATEMENT, IN TURN, MAY CONTAIN SYMBOLS, OPERATIONS, CONSTANTS, BRACKETS, STATEMENT SYMBOLS, AND REGION SYMBOLS. RATHER THAN EXPLAIN ALL OF THESE TERMS AT ONCE, EXAMPLES WILL BE USED TO INTRODUCE THESE CONCEPTS AS MORE INTRICATE PROBLEMS ARE PRESENTED.

SYMBOLS AND OPERATIONS

CONSIDER THE FOLLOWING PROBLEM.

rate x time : dist.

THIS PROBLEM AS STATED COULD BE ENTERED INTO THE LGP-30 EXACTLY AS IT APPEARS AND THE ACT I PROGRAM WOULD MAKE AN "OBJECT" PROGRAM OF IT. THE PROBLEM STATEMENT IS THEN SUITABLE AS A "SOURCE" STATEMENT. ON EXAMINATION OF THE "SOURCE" PROGRAM IN MORE DETAIL WE FIND THAT IT IS MADE UP OF SYMBOLS (rate, time, dist.) AND OPERATIONS (x, :).

A SYMBOL IS DEFINED IN THE ACT I SYSTEM AS BEING ANY FIVE OR LESS ALPHANUMERIC CHARACTERS ON THE LGP-30 TAPEWRITER WITH THE FOLLOWING RESTRICTIONS.

1. IT CANNOT BE AN OPERATION (SEE TABLE - FIGURES 3 AND 4)
2. IT CANNOT BE A CONSTANT 0-99999
3. IT CANNOT BE A BRACKET
4. IT CANNOT BE A STATEMENT SYMBOL (SEE DEF. PG. 3)
5. IT CANNOT BE A REGION SYMBOL (SEE DEF. PG. 3)

THIS MAY APPEAR TO PLACE MANY RESTRICTIONS UPON A SYMBOL, BUT IT IS DOUBTFUL IF ANYONE WILL HAVE ANY TROUBLE REMEMBERING THESE RULES. IT IS QUITE UNLIKELY THAT SOMEONE WOULD WANT TO WRITE AS A SYMBOL

sin + or a +]

PROVISION HAS BEEN MADE FOR 127 SYMBOLS. IF MORE ARE REQUIRED, A REGION MAY BE USED FOR THAT PURPOSE (SEE REGION).

ANY OF THE OPERATIONS THAT ARE GIVEN IN THE TABLES MAY BE USED IN STATEMENTS. THESE TABLES ARE FLEXIBLE AND CAN BE EXPANDED AND/OR CHANGED VERY EASILY. SOME OF THE OPERATIONS MAKE THE ACT I SYSTEM EXTREMELY POWERFUL AND VERSATILE OPERATIONS ARE EXPLAINED IN A SEPARATE SECTION. PROVISIONS HAS BEEN MADE FOR 63 OPERATIONS.

BRACKETS AND CONSTANTS

CONSIDER THE FOLLOWING PROBLEM AND THE STATEMENT REQUIRED TO EVALUATE IT.

$$\frac{3 + ab}{b^2 - 4ac} = y$$

THIS WOULD BE PRESENTED TO THE ACT I SYSTEM IN THE FOLLOWING "SOURCE" LANGUAGE.

[3 + a x b]/[b x b - 4 x a x c] : y

THIS STATEMENT AGAIN CONTAINS SYMBOLS (a, b, c, y) AND OPERATIONS (+, - x, /, :). IN ADDITION TO THESE TERMS IT CONTAINS BRACKETS ([,]) AND CONSTANTS (3, 4).

THE BRACKETS ARE FAIRLY SELF-EXPLANATORY; THEY ARE USED TO ENCLOSE A PORTION OF A STATEMENT. IT MAY BE NOTED THAT BRACKETS MIGHT BE REQUIRED MORE OFTEN IN THE ACT I SYSTEM THAN IN HANDWRITTEN ALGEBRA BECAUSE THE TAPE TYPEWRITER DOES NOT HAVE SPECIAL CHARACTERS LIKE $\sqrt{}$ AND FURTHERMORE, THE STATEMENT MUST BE PRESENTED IN ONE LINE OF TYPE. BRACKETS MAY BE NESTED SIX DEEP.

BRACKETS WILL HAVE SOME EFFECT ON THE TIME SPENT TO COMPILE THE "OBJECT" PROGRAM, BUT THE LENGTH OF THE "OBJECT PROGRAM AND ITS TIME OF EXECUTION WILL NOT BE AFFECTED. SO IF THERE IS EVER ANY DOUBT IF A BRACKET SHOULD BE USED OR NOT; USE IT.

THE CONSTANTS (3 AND 4) ARE SELF-EXPLANATORY. THEY MAY BE ANY FIVE DIGIT POSITIVE NUMBERS. PROVISION HAS BEEN MADE FOR 39 CONSTANTS.

STATEMENT SYMBOLS

ANOTHER TYPE PROBLEM THAT FREQUENTLY ARISES IS WHEN A LOGICAL DECISION MUST BE MADE IN A PROGRAM. IN THE FOLLOWING EXAMPLE IT IS DESIRED TO EVALUATE A PRESSURE IN THREE DIFFERENT MANNERS DEPENDING ON WHETHER ("A" GREATER 50), ("A" LESS 50), OR ("A" EQUAL 50). THE "SOURCE" LANGUAGE FOR THE ACT I SYSTEM TO MAKE THIS DECISION AND TRANSFER TO THE APPROPRIATE PROGRAM WOULD BE AS FOLLOWS:

```
when a less 50 trn sl2

when a grt 50 trn sl3

a x a - b : pres  use s25

sl2 a x a x a : pres use s25

sl3 a - b x b : pres

s25 pres x (.....
```

THIS STATEMENT AGAIN CONTAINS SYMBOLS (a, b, pres), OPERATIONS (less, trn, when grt, x, -, :, use) AND A CONSTANT 50. NO BRACKETS WERE REQUIRED.

STATEMENT SYMBOLS HAVE BEEN GIVEN TO THE FOURTH, FIFTH, AND SIXTH STATEMENTS. IN THE CASE OF THE FIRST STATEMENT, THE STATEMENT SYMBOL sl2 IS USED TO PROVIDE A STATEMENT TO TRANSFER TO IN THE EVENT THE VALUE FOR SYMBOL "A" IS LESS THAN 50. IF THE VALUE FOR "A" IS NOT LESS THAN 50, THE TRANSFER TO STATEMENT STARTING WITH sl3 DOES NOT OCCUR AND THE NEXT SEQUENTIAL STATEMENT IS EXECUTED.

STATEMENT 2 THEN INTERROGATES TO SEE IF "A" IS GREATER THAN 50. IF IT IS, THE STATEMENT STARTING WITH sl3 IS EXECUTED; IF NOT THE LATTER CASE, STATEMENT 3 IS EXECUTED. THIS CAN ONLY HAPPEN IF "A" IS NEITHER LESS THAN 50 NOR GREATER THAN 50 - THAT IS IF "A" IS EQUAL TO 50.

THE "use s25" PART IN THE THIRD AND FOURTH STATEMENTS UNCONDITIONALLY TRANSFERS TO THE STATEMENT STARTING WITH s25. THE PROGRAM THAN CALCULATES THE VALUE FOR SYMBOL "pres" IN THREE DIFFERENT MANNERS DEPENDING IF "A" LESS 50, "A" GREATER 50, OR "A" EQUAL 50. ALL PATHS OF CALCULATION COME BACK TOGETHER AT STATEMENT LABELED WITH STATEMENT SYMBOL s25.

THE STATEMENT SYMBOLS MAY BE s0 THROUGH s255. PROVISION HAS BEEN MADE FOR 256 STATEMENT SYMBOLS WHICH MUST START WITH AN "s" FOLLOWED BY 1 TO 3 DIGITS WHICH MUST BE LESS THAN 256.

REGION SYMBOLS

WHEN ARRAYS OR SETS OF NUMBERS ARE REQUIRED DURING A CALCULATION, THE COMMON LANGUAGE OF ALGEBRA BEGINS TO BECOME INADEQUATE UNLESS MATRIX ALGEBRA METHODS ARE ADOPTED. THE MATRIX ALGEBRA TERMINOLOGY DOES NOT ENJOY THE SAME RECOGNITION AS COMMON ALGEBRA AND FURTHERMORE THE TAPE TYPEWRITER IS NOT ADAPTED TO EXPRESS IT.

NEVERTHELESS, PROBLEMS CONTAINING ARRAYS OR SETS OF NUMBERS STILL EXIST. CONSIDER THE PROBLEM WHERE IT IS DESIRED TO CALCULATE THE SUM OF 500 NUMBERS. THE STATEMENTS TO SOLVE THIS PROBLEM COULD BE AS FOLLOWS.

```
dim a 501
index i
0 : sum
1 : i
sl a i + sum : sum
iter i 1 500 sl
stop
```

THIS PROGRAM OF STATEMENTS LOOKS MORE INVOLVED THAN THE FIRST; HOWEVER, THERE IS ONLY ONE NEW TERM INTRODUCED IN IT. THE STATEMENTS STILL CONTAIN SYMBOLS (i sum) OPERATIONS (dim, index, :, +, stop, iter), CONSTANTS (0, 1, 500) AND THE STATEMENT SYMBOL (s1).

THE REGION SYMBOL "A" HAS BEEN ASSIGNED TO REFER TO A BLOCK OF 501 LOCATIONS. THE FIRST STATEMENT

dim a 501

RESERVES 501 LOCATIONS OR WORDS, WHICH WILL BE DESIGNATED BY THE REGION SYMBOL "A". IF THE SELECTION OF ANY WORD IN THE BLOCK IS DESIRED, A CONSTANT OR A SYMBOL WHICH IS ASSIGNED AS AN INDEX REGISTER MUST FOLLOW THE REGION SYMBOL.

THE FIRST WORD IN A REGION IS WORD NUMBER 0. IN THE EXAMPLE 500 NUMBERS ARE TO BE ADDED. SINCE THE INDEX i HAS BEEN SET TO 1, 501 MEMORY LOCATIONS MUST BE RESERVED (THE FIRST WORD, WORD NUMBER 0, IS NOT USED IN THE SUM).

EXAMPLES:

dim a 500 index i j	SET i AND j AS INDICES
a 0 + b	1ST WORD OF REGION "A" ADDED TO B
a 14 + c	15TH WORD OF REGION "A" ADDED TO C
a i + a j	ITH (i + 1) TH WORD OF REGION "A" ADDED TO THE (j + 1)th WORD OF REGION "A". IF THE FIRST WORD, WORD NUMBER 0, OF A REGION IS IGNORED, AS IN THE SUMMATION EXAMPLE, a i + a j WILL REPRESENT THE ith WORD ADDED TO THE jth WORD.

A REGION SYMBOL IS ANY SYMBOL THAT HAS BEEN DEFINED AS SUCH BY A DIMENSION STATEMENT (SEE PAGE 5). WHEN USED IN A STATEMENT IT MUST BE FOLLOWED BY A CONSTANT OR SYMBOL FOR AN INDEX REGISTER. PROVISION HAS BEEN MADE FOR 11 REGION SYMBOLS.

IN THE PREVIOUS EXAMPLE (SHOWN ON PAGE 3) EACH OF THE STATEMENTS IS EXPLAINED AS FOLLOWS:

dim a 501	SET ASIDE 501 MEMORY LOCATIONS WHICH WILL BE REFERRED TO BY REGION SYMBOL "A".
index i	SET "i" AS A SYMBOL WITH THE SPECIAL FUNCTION OF AN INDEX
0 : sum	SET THE VALUE FOR SYMBOL "SUM" EQUAL TO 0.
1 : i	SET THE VALUE FOR SYMBOL "i" EQUAL TO 1.
s1 a i + sum: sum	TAKE THE ITH VALUE OF REGION "A" (1ST TIME THRU IT WILL BE A 1) AND ADD IT TO "SUM" (1ST TIME THRU IT WILL BE 0), AND STORE THE RESULT IN "SUM"
iter i 1 500 s1	INCREASE "i" BY 1. IF "i" IS LESS THAN OR EQUAL TO 500, TRANSFER TO STATEMENT STARTING WITH s1. IF "i" IS GREATER THAN 500, GO ON TO NEXT STATEMENT
stop	STOP COMPUTATION.

III SPECIAL STATEMENT FORMS

THERE ARE FOUR OPERATIONS LISTED IN THE OPERATION TABLE THAT REQUIRE A SPECIAL FORMAT WHEN THEY ARE USED IN A STATEMENT. WHEN ANY OF THESE FOUR OPERATIONS ARE USED IN A STATEMENT, IT MUST BE THE ONLY OPERATION IN THAT STATEMENT.

dim - DIMENSION

DIM STATEMENTS MUST PRECEDE ALL OTHER STATEMENTS. THEIR FUNCTION IS TO RESERVE A BLOCK OF MEMORY TO BE USED AS A REGION. EACH DIM STATEMENT WILL RESERVE A BLOCK OF MEMORY AND ASSOCIATE A REGION SYMBOL TO THAT BLOCK, I.E.

```
dim abc 500
```

THIS STATEMENT WILL RESERVE 500 MEMORY LOCATIONS FOR A REGION DESIGNATED abc THE REGION SYMBOL abc IMMEDIATELY FOLLOWS THE OPERATION dim. THE REGION SYMBOL MAY NOT BE USED FOR ANY PURPOSE EXCEPT TO REFER TO THIS REGION. THERE MAY BE 11 dim STATEMENTS EACH USING A DIFFERENT REGION SYMBOL. THE dim STATEMENT CREATES NO INSTRUCTIONS IN THE "OBJECT" PROGRAM.

index - SET INDEX REGISTERS

AN index STATEMENT IF USED MUST IMMEDIATELY FOLLOW THE dim STATEMENTS. THERE MUST BE ONLY ONE index STATEMENT. AS MANY SYMBOLS AS DESIRED MAY BE USED FOR INDEX REGISTERS, HOWEVER, THEY MUST ALL BE LISTED IN THE SAME index STATEMENT, I.E.

```
index i j k rate
```

THIS STATEMENT WILL USE (i j k rate) FOR INDEX REGISTERS. THESE SYMBOLS, OF COURSE, CANNOT HAVE BEEN USED FOR REGION SYMBOLS. FOR EACH SYMBOL IN THE STATEMENT, THREE INSTRUCTIONS ARE CREATED IN THE "OBJECT" PROGRAM.

iter - ITERATE

THE iter STATEMENT IS USED TO INCREASE THE VALUE FOR A SYMBOL BY A FIXED AMOUNT AND TEST IT AGAINST A LIMIT. FREQUENTLY, THE SYMBOL INCREASED WOULD BE ASSIGNED AS AN INDEX REGISTER, I.E.

```
iter i 1 10 s11
```

THIS STATEMENT WILL INCREASE THE VALUE OF SYMBOL i BY 1. IF THE NEW VALUE OF i IS LESS THAN OR EQUAL TO 10 THE NEXT STATEMENT TO BE EXECUTED WILL BEGIN WITH STATEMENT SYMBOL s11. IF THE NEW VALUE OF i IS GREATER THAN 10, THE NEXT STATEMENT TO BE EXECUTED WILL BE THE STATEMENT FOLLOWING THE ITERATION STATEMENT. EITHER THE 1 OR 10 MAY BE A SYMBOL INSTEAD OF A CONSTANT, I.E.

```
or  iter i 1 ab s14
    iter j i k s27
```

THERE IS NO LIMIT ON THE NUMBER OF iter STATEMENTS THAT MAY BE USED. EACH STATEMENT WILL CREATE SIX INSTRUCTIONS IN THE "OBJECT" PROGRAM.

sub - SUBROUTINE

THE sub STATEMENT WILL ENABLE THE USER OF THE ACT I SYSTEM TO USE SUBROUTINES THAT ARE NOT INCORPORATED IN THE BASIC TABLE OF OPERATIONS. THE USE OF THIS IS BEST ILLUSTRATED WITH TWO EXAMPLES.

1. ASSUME THE PROGRAMMER WANTED TO USE A HYPERBOLIC TANGENT SUBROUTINE AND THIS ROUTINE IS NOT INCLUDED IN THE BASIC TABLE OF OPERATIONS. THE PROGRAMMER WOULD FIRST SET ASIDE A REGION IN WHICH HE COULD PLACE THE HYPERBOLIC TANGENT ROUTINE, I.E.

   ```
   dim tanh 173
   ```

   WHEN THE PROGRAMMER WISHED TO USE THIS ROUTINE TO EVALUATE THE HYPERBOLIC TANGENT OF xy, AND STORE THE RESULT txy THE STATEMENT WOULD BE AS FOLLOWS: I.E.

   ```
   sub tanh xy txy
   ```

2. IF TWO MATRICES a AND b ARE TO BE MULTIPLIED AND THE RESULT STORED IN c AND ASSUMING THE MATRIX MULTIPLY SUBROUTINE REQUIRED THE INITIAL LOCATIONS OF a, b, AND c AND THEIR DIMENSIONS, THE sub STATEMENT COULD LOOK AS FOLLOWS: I.E.

   ```
   sub matx a b c i j k
   ```

IV EXAMPLES

SEVERAL EXAMPLES HAVE BEEN USED IN THE INTRODUCTION AND DEFINITION SECTIONS OF THIS MANUAL. THE FOLLOWING EXAMPLES ARE USED TO FURTHER ILLUSTRATE THE PROGRAMMING TECHNIQUES WITH SPECIAL EMPHASIS ON INDEXING METHODS.

THE FIRST EXAMPLE SHOWS THE PROGRAMMING REQUIRED TO CALCULATE THE VARIANCE OF A SET OF NUMBERS. THE SET MAY CONTAIN ANY N NUMBER OF VALUES WITH THE RESTRICTION N IS LESS THAN 1000. AFTER THE DIM AND INDEX STATEMENTS THE VALUE OF N IS READ INTO THE COMPUTER. THE PROGRAM THEN CONSISTS OF THREE LOOPS:

1. READING THE N VALUES OF A INTO THE LGP-30
2. CALCULATING THE AVERAGE OF THE SET "A" = $\bar{A}$ = ABAR.
3. CALCULATING THE VARIANCE.

WITH A LITTLE MORE JUDICIOUS STUDY OF THE PROBLEM, THE PROGRAMMER WOULD PROBABLY COMBINE THE FIRST TWO LOOPS. CALCULATING THE AVERAGE VALUE OF "A" IN THE SAME LOOP AS READING "A" INTO THE COMPUTER WOULD RESULT IN FEWER PROGRAM STEPS AND A FASTER RUNNING PROGRAM. THE PROGRAM BELOW WILL USE THE THREE LOOPS AS FIRST OUTLINED.

NOTE THESE TWO POINTS:

1. THE : OPERATION CAN BE USED INSIDE A STATEMENT. (SEE SECTION ON BRACKETS AND PRECEDENCE)
2. 1000 LOCATIONS ARE SET ASIDE FOR REGION A BUT THE a 0 IS NOT USED. HENCE, ONLY 999 VALUES OF a, I.E. a 1 THRU a 999 MAY BE USED.

VARIANCE	VAR = $\frac{1}{N}\sum (A - \bar{A})^2$
dim a 1000	RESERVE 1000 LOCATIONS FOR REGION "A"
index i	USE I AS AN INDEX
s1 read n	READ N - NUMBER OF VALUES IN REGION "A"
1 : i s10 read a i iter i 1 n s10	READ IN THE N VALUES OF "A"
1 : i 0 : sum s2 a i + sum : sum iter i 1 n s2 sum / n : abar	CALC. $\bar{A}$ = ABAR
1 : i 0 : sum s3 [a i - abar : s] x s + sum : sum iter i 1 n s3 sum / n : var	CALC. VARIANCE = VAR
0 print abar 0 print var use s1	

FIGURE 1

THE SECOND EXAMPLE SHOWS A PROGRAMMING METHOD FOR A SIMPLIFIED ACCOUNTING FUNCTION, THAT IS, THE DISTRIBUTION OF COSTS AND PIECES PRODUCED FOR THE PAYROLL AND JOB COSTING DEPARTMENTS. IN THIS EXAMPLE, WE ASSUME THERE ARE 20 DEPARTMENTS, 300 EMPLOYEES, AND 500 JOBS. THESE ARE ASSIGNED SEQUENTIAL NUMBERS AND ARE GIVEN THE NAMES OF DEPT, CLOCK, AND JOB.

FILES MUST BE SET UP FOR THE VARIOUS INFORMATION THAT IS TO BE ACCUMULATED AND USED. THEY ARE:

FILE	SIZE	SYMBOL USED
DEPARTMENT COST	20	DPT C
EMPLOYEE EARNINGS	300	EARN
EMPLOYEE RATES	300	RATE
JOB HOURS	500	JOB H
JOB COSTS	500	JOB C
JOB PIECES	500	JOB P

THE EMPLOYEE WILL GIVE TO THE ACCOUNTING DEPARTMENT TICKETS FOR EACH JOB HE HAS WORKED ON DURING THE DAY. THESE TICKETS WILL CONTAIN THE FOLLOWING INFORMATION:

DEPARTMENT	DEPT
EMPLOYEE NUMBER	CLOCK
JOB NUMBER	JOB
HOURS WORKED	HOURS
PIECES PRODUCED	PIECES

THE APPROACH USED WILL BE TO SET UP THE GIVEN FILES WITH DIM STATEMENTS AND USE DEPT, CLOCK AND JOB AS INDICES (SOMETIMES CALLED KEYS) TO SELECT THE PARTICULAR ITEM DESIRED FROM EACH FILE. FOR INSTANCE, IN STATEMENT s3 THE EMPLOYEE'S CLOCK NUMBER CLOCK IS USED TO SELECT HIS RATE FROM THE RATE FILE AND THIS IS MULTIPLIED BY THE HOURS WORKED FOR THAT TICKET.

THIS WILL GIVE A COST FOR THAT TICKET WHICH IS TO BE DISTRIBUTED INTO THE EMPLOYEE EARNINGS, THE DEPARTMENT COST, AND THE JOB COST.

THERE ARE NATURALLY MANY MODIFICATIONS WHICH CAN BE MADE TO A PROGRAM OF THIS TYPE. IN THIS PROGRAM, A PRINT OF ALL THE UPDATED FILE INFORMATION WILL OCCUR IF AN ARTIFICIAL DEPARTMENT NUMBER (DEPT. = 30) IS GIVEN. THE PRINT OUT OF THE PROGRAM STARTS AT STATEMENT s20.

COSTS AND DISTRIBUTIONS

```
dim dpt c 20
dim earn 300
dim rate 300
dim job h 500
dim job c 500
dim job p 500

index clock dept job i
sl read dept                                   READ TICKET

when dept equal 30 trn s20

read clock
read job
read hours
read pcs.

rate clock x hours : cost
cost + earn clock : earn clock                 DISTRIBUTE COST TO EARNINGS
cost + dpt c dept : dpt c dept                 DISTRIBUTE COST TO DEPARTMENT
cost + job c job : job c job                   DISTRIBUTE COST TO JOB
hours + job h job : job h job                  DISTRIBUTE HOURS TO JOB
pcs. + job p job : job p job                   DISTRIBUTE PIECES PRODUCED TO JOB
use sl                                         READ ANOTHER TICKET

s20 1 : i
s2 0 print i 2 print dpt c i cr
iter i 1 20 s2

cr cr                                          2 CARRIAGE RETURNS
1 : i
s3 0 print i 2 print earn i cr
iter i 1 300 s3

cr cr                                          2 CARRIAGE RETURNS
1 : i
s4 0 print i 1 print job h i 2 print job c i 0 print job p i cr
iter i 1 500 s4
stop
```

FIGURE 2

SOME OF THE INDEX SYMBOLS ARE UNDERLINED IN THIS EXAMPLE FOR THE SAKE OF CLARITY.

V OPERATIONS

THE OPERATIONS IN TABLE I AND II (FIGURES 3 & 4) REPRESENT A SET OF OPERATIONS THAT MAY BE USED WITH THE ACT I SYSTEM. THE OPERATION SYMBOLS MAY BE CHANGED, DELETED, OR OTHER OPERATIONS MAY BE ADDED TO THESE TABLES.

THERE IS NO ABSOLUTE DISTINCTION BETWEEN TABLE I AND II (FIGURES 3 & 4) EXCEPT FOR THE FACT THAT TABLE II (FIGURE 4) DEALS PRIMARILY WITH MACHINE LANGUAGE OPERATION AND THE USE OF THESE OPERATIONS MIGHT BE CALLED USING A SYMBOLIC ASSEMBLY PROGRAM RATHER THAN A COMPILER. THE OPERATIONS IN TABLE I (FIGURE 3) INCLUDE INPUT, OUTPUT, AND BASIC ARITHMETIC IN FIXED AND FLOATING POINT AND THE SPECIAL STATEMENT OPERATIONS AS PREVIOUSLY DEFINED.

TABLE OF OPERATIONS I

OPERATION CODE	USE	MEANING
:	a : b	SET VALUE FOR "B" EQUAL TO VALUE FOR "A"
+	a + b	"B" ADDED TO "A"
–	a – b	"B" SUBTRACTED FROM "A"
x	a x b	"B" MULTIPLIED BY "A"
/	a / b	"B" DIVIDED INTO "A"
when		
less	when a less b trn sl05	A < B GO TO STATEMENT sl05
grt	when a grt b trn s40	A > B GO TO STATEMENT s40
equal	when a equal b trn sl	A = B GO TO STATEMENT sl
trn	trn sl7	USED AS ABOVE
abs	abs a	|A|
read	read a	READ A VALUE FOR SYMBOL "A"
print	n print a	PRINT THE VALUE OF SYMBOL "A" WITH N DEC.PLACES
stop	stop	STOP COMPUTATION
f+	a f+ b	A + B IN FLOATING ARITHMETIC
f–	a f– b	A – B IN FLOATING ARITHMETIC
fx	a fx b	A X B IN FLOATING ARITHMETIC
f/	a f/ b	A / B IN FLOATING ARITHMETIC
finp	finp a	READ IN FLOATING VALUE FOR SYMBOL "A"
fprt	fprt a	PRINT THE FLOATING VALUE OF SYMBOL "A"
flo	n flo a	CONVERT THE VALUE OF "A" FROM FIXED TO FLOATING POINT. "A" HAD N DECIMAL PLACES
unflo	n unflo a	CONVERT THE VALUE OF "A" FROM FLOATING TO FIXED POINT. "A" WILL HAVE N DECIMAL PLACES
iter	iter k i n sl4	SEE SPECIAL STATEMENT SECTION III
dim	dim a 500	SEE SPECIAL STATEMENT SECTION III
index	index i j k	SEE SPECIAL STATEMENT SECTION III
sub	sub tanh xy txy	SEE SPECIAL STATEMENT SECTION III
tab	tab	CARRIAGE TAB
cr	cr	CARRIAGE RETURN
ret	ret s4	SET RETURN ADDRESS IN STATEMENT S4. STATEMENT S4 MUST BE OF FORM s4 use S00
use	use s8	UNCONDITIONAL TRANSFER TO STATEMENT S8.

FIGURE 3

TABLE II

OPERATION CODE	USE	MEANING
bring	bring a	REPLACE THE CONTENTS OF THE ACCUMULATOR WITH THE CONTENTS OF THE MEMORY LOCATION SPECIFIED BY THE ADDRESS, A.
add	add a	ADD THE CONTENTS OF A TO THE CONTENTS OF THE ACCUMULATOR AND RETAIN THE SUM IN THE ACCUMULATOR.
subtr	subtr a	SUBTRACT THE CONTENTS OF A FROM THE CONTENTS OF THE ACCUMULATOR AND RETAIN THE DIFFERENCE IN THE ACCUMULATOR.
mult	mult a	MULTIPLY THE NUMBER IN THE ACCUMULATOR BY THE NUMBER IN MEMORY LOCATION, A, RETAINING THE MOST SIGNIFICANT HALF OF THE PRODUCT IN THE ACCUMULATOR.
nmult	nmult a	MULTIPLY THE NUMBER IN THE ACCUMULATOR BY THE NUMBER IN MEMORY LOCATION, A, RETAINING THE LEAST SIGNIFICANT HALF OF THE PRODUCT IN THE ACCUMULATOR.
div	div a	DIVIDE THE NUMBER IN THE ACCUMULATOR BY THE NUMBER IN THE MEMORY LOCATION, A, RETAINING THE QUOTIENT.
extrt	extrt a	THE EXTRACT ORDER OR "LOGICAL PRODUCT" IS A BIT BY BIT PRODUCT OF THE CONTENTS OF THE ACCUMULATOR BY THE CONTENTS OF A.
hold	hold a	STORE THE CONTENTS OF THE ACCUMULATOR IN A, RETAINING THE CONTENTS OF THE ACCUMULATOR IN THE ACCUMULATOR.
clear	clear a	STORE THE CONTENTS OF THE ACCUMULATOR IN MEMORY LOCATION A, CLEARING THE ACCUMULATOR TO ZERO.
stadd	stadd a stadd s17	STORE ONLY THE ADDRESS PORTION OF THE WORD IN THE ACCUMULATOR IN MEMORY LOCATION A, LEAVING THE REST OF THE WORD IN A UNDISTURBED.
ret	ret a ret s7	ADD "ONE" TO THE ADDRESS HELD IN THE COUNTER REGISTER, C, AND RECORD IT IN THE ADDRESS PORTION OF THE WORD IN MEMORY LOCATION, A. THE COUNTER REGISTER C, NORMALLY CONTAINS THE ADDRESS OF THE NEXT INSTRUCTION TO BE EXECUTED. THIS COMMAND IS USED IN SETTING A SUB-ROUTINE EXIT.
use	use a use s4	TRANSFER CONTROL TO A UNCONDITIONALLY, I.E., GET THE NEXT INSTRUCTION FROM MEMORY LOCATION, A.
trn	trn a trn s114	CONDITIONAL TRANSFER, TRANSFER CONTROL TO MEMORY LOCATION A, ONLY IF THE NUMBER IN THE ACCUMULATOR IS NEGATIVE. OTHERWISE, THE TEST COMMAND IS IGNORED.
rdhex	rdhex a	READ A HEXIDECIMAL VALUE FOR SYMBOL A.

FIGURE 4

VI USE

1. RULES FOR PUNCHING "SOURCE" PROGRAM TAPE.

   A. A CONDITIONAL STOP (') MUST FOLLOW EACH SYMBOL, OPERATION, CONSTANT, BRACKET, STATEMENT SYMBOL, AND REGION SYMBOL.

   B. A CONDITIONAL STOP (') MUST FOLLOW EACH STATEMENT.

   C. ANOTHER CONDITIONAL STOP (') MUST FOLLOW THE FINISHED "SOURCE" LANGUAGE PROGRAM, I.E.

```
a'+'b':'c''
O'print'c'''
```

   D. THE LGP-30 MAKES NO DISTINCTION BETWEEN UPPER AND LOWER CASE ON THE TYPEWRITER, THEREFORE:

```
:  IS THE SAME AS ;
?  IS THE SAME AS /
[  IS THE SAME AS ,
]  IS THE SAME AS .
     ETC.
```

   THE PROGRAMMER MAY USE THE CHARACTER WHICH IS EASIER OR MORE APPEALING TO HIM.

2. PROCEDURE TO COMPILE A PROGRAM.

   A. LOAD THE ACT I SYSTEM INTO THE LGP-30 WITH THE PROGRAM INPUT ROUTINE (10.4)

      1. COMPILER (HEX TAPE)
      2. OPERATION CODES (HEX TAPE)

   B. PLACE THE "SOURCE" PROGRAM TAPE INTO THE READER.

   C. TRANSFER TO 4000 TO READ AND COMPILE THE "SOURCE" PROGRAM.

   D. DEPRESS SIX BIT INPUT BUTTON.

   E. AFTER CREATING THE "OBJECT" PROGRAM, THE ACT I SYSTEM PRINTS THE FOLLOWING INFORMATION.

      1. INITIAL LOCATION TO TRANSFER TO EXECUTE THE "OBJECT" PROGRAM

         i 0322

      2. FINAL LOCATION OF THE "OBJECT" PROGRAM. f 0734

      3. STATEMENT SYMBOLS AND THE LOCATION ASSIGNED TO EACH.

3. PUNCH OUT PROCEDURE
   THE COMPUTER WILL STOP AFTER PRINTING THE STATEMENT SYMBOL TABLES. RAISE THE TRANSFER CONTROL BUTTON. TO PUNCH THE "OBJECT" PROGRAM AND NECESSARY INFORMATION FROM THE LGP-30, TURN THE PUNCH ON AND DEPRESS THE START BUTTON. IF HIGH-SPEED PUNCH IS USED, DEPRESS BREAKPOINT BUTTON #32.

4. PROCEDURE TO EXECUTE COMPILED PROGRAM.

   A. LOAD THE SUBROUTINE TAPE INTO THE LGP-30 WITH THE PROGRAM INPUT ROUTINE (10.4)

   B. LOAD THE TAPE FROM (3) INTO THE LGP-30 WITH THE PROGRAM INPUT ROUTINE (10.4) THIS IS NOT NECESSARY IF PROGRAM WAS JUST COMPILED).

5. PROCEDURE TO CHECKOUT PROGRAM.

   A. DEPRESS TRANSFER CONTROL BUTTON ON LGP-30 CONSOLE BEFORE TRANSFERRING TO 4000 TO READ AND COMPILE THE "SOURCE" PROGRAM. THIS WILL ADD TWO INSTRUCTIONS AT THE END OF EACH STATEMENT WHICH WILL BE EXECUTED DURING "COMPUTE" TIME.

   B. DURING "COMPUTE" TIME, A PROGRAM COMPILED UNDER THE ABOVE CONDITION WILL BEHAVE IN THE FOLLOWING MANNER.

      1. TRANSFER CONTROL BUTTON UP- NO EFFECT.

      2. TRANSFER CONTROL BUTTON DOWN -

A. THE LOCATION OF THE FIRST INSTRUCTION IN THE NEXT STATEMENT WILL BE PRINTED. THIS LOCATION CAN BE COMPARED WITH THE PRINT OUT OF THE STATEMENT SYMBOLS AND THEIR CORRESPONDING LOCATIONS, TO DETERMINE WHICH STATEMENT IS BEING COMPUTED.

B. THE RESULTS OF THE LAST OPERATION PERFORMED IN THE STATEMENT WILL BE PRINTED.

6. THE FOLLOWING ERROR CODES INDICATE THE TYPE OF ERROR THAT HAS BE N DETECTED BY THE ACT I SYSTEM AND ANY CORRECTIVE ACTION THAT MAY BE TAKEN.

ERROR CODE	MEANING	CORRECTIVE ACTION
e 1	6-BIT BUTTON IS UP	DEPRESS 6-BIT BUTTON AND START
e 2	PROGRAM TOO LARGE - OVERLAPS REGIONS	CORRECT PROGRAM AND RESTART
e 3	VALUE IN DIM STATEMENT NOT NUMBER	CORRECT STATEMENT AND RESTAR
e 4	TWO INDEX STATEMENTS	CORRECT PROGRAM AND RESTART
e 5	BRACKET COUNT IS IN ERROR	CORRECT STATEMENT AND RESTART
e 6	STATEMENT IS TOO LARGE	CORRECT STATEMENT AND RESTART
e 7	STATEMENT SYMBOL NOT DEFINED	CORRECT PROGRAM AND RESTART
e 8	INCORRECT REGIONAL DESIGNATION	CORRECT STATEMENT AND RESTART
e 9	INCORRECT LEFT OPERAND	CORRECT STATEMENT AND RESTART

FIGURE 5

7. PROGRAMMING LIMITATIONS.

1. NO OPERATION IS TO BE ASSUMED. BRACKETS DO NOT IMPLY MULTIPLICATION.

2. THERE IS A MAX.OF 63 STOP CODES IN ANY STATEMENT.

3. THERE MUST BE AN EQUAL NUMBER OF FRONT AND BACK BRACKETS IN EACH STATEMENT.

4. EACH STATEMENT SYMBOL USED IN A STATEMENT MUST BE USED AT THE BEGINNING OF SOME STATEMENT.

5. IF A REGION SYMBOL IS USED IN ANY STATEMENT (EXCEPT A SUB STATEMENT), IT MUST BE FOLLOWED BY A CONSTANT OR A SYMBOL WHICH IS ASSIGNED AS AN INDEX REGISTER.

6. A TYPEWRITER "SPACE" MAY BE USED AS ANY OF THE OTHER ALPHABETICAL CHARACTERS. IT MUST NOT BE USED TO SEPARATE SYMBOLS, OPERATIONS, ETC.

7. ALL INDEXED VARIABLES (REGIONS) MUST APPEAR IN DIMENSION STATEMENTS.

8. ALL OPERATIONS ARE IN THE NUMERATOR WITH THE EXCEPTION OF THE SYMBOL OR GROUPING ENCLOSED IN BRACKETS IMMEDIATELY FOLLOWING THE DIVISION OPERATION.

9. AT LEAST ONE OPERATION SHOULD BE IN EACH SET OF BRACKETS.

10. ALGEBRAIC OPERATIONS SHOULD NOT BE MIXED WITH DECISION OR PRINT OPERATIONS UNLESS THE ALGEBRA IS BRACKETED.

11. THE FIRST STATEMENT SHOULD BE

    dim'comp'512''

    THIS STATEMENT WILL RESERVE 512 LOCATIONS FOR THE SUBROUTINE PACKAGE. IF THE SUBROUTINE PACKAGE IS RELOCATED OR ADDITIONS ARE MADE, THE NUMBER OF LOCATIONS TO RESERVE SHOULD REPLACE 512. THE NUMBER OF LOCATIONS TO RESERVE IS THE NUMBER OF LOCATIONS BETWEEN THE FIRST ONE TO RESERVE AND LOCATION 5800.

VII APPENDIX

1. Scaling

The ACT I System will generate a fixed and floating point program. It is the responsibility of the programmer to keep these two formats separate.

A. Floating Point

Scaling as such is no problem when all arithmetic is done in the floating point mode. The only concern to the programmer is the format of the numbers that are to be presented to the computer. Floating Point numbers are entered in the following manner.

NUMBER	PRESENTED AS
123.45	1234500' 03'
.000781	7810000' -03'
-.123	-1230000' 00'

NOTE:

1. Seven (7) digits are always entered in the mantissa.
2. If the number is negative, the (-) minus sign precedes the 7 digits.
3. A conditional stop (') follows the mantissa.
4. The characteristic may vary between 30 to 0 and -01 to -30.
5. A conditional stop (') follows the characteristic.

B. Fixed Point

The primary purpose for fixed point arithmetic in the ACT I System is for the manipulation of the index Symbols. There will arise cases, such as the two examples in this manual, where fixed point arithmetic will be more desirable than floating point.

To the ACT I System all fixed point numbers are thought of as integers. Any scaling is done externally by the programmer who decides arbitrarily how many fractional positions will be carried to the right of the decimal point. I.E., the computer might contain 1378 as an integer, but the programmer may wish to think of this as 1.378.

In this case he is carrying the number with 3 decimal places.

The programmer may effectively change the decimal position by multiplying or dividing the number by a power of 10.

I.E., a HAS 3 DECIMAL PLACES
b HAS 2 DECIMAL PLACES

TO ADD:

a + (b x 10) : c

To carry this example further

a/b : d AT 1 DECIMAL PLACE
axb : e AT 5 DECIMAL PLACES

The following statements will print the values of (c, d, e) with their proper decimal point.

```
3 print c
1 print d
5 print e
```

A FEW NOTES OF CAUTION ON FIXED POINT PROGRAMMING:

1. THE MAXIMUM INTEGER THAT CAN BE DEVELOPED THROUGH MULTIPLICATION IS 134 217 727 REGARDLESS OF SCALING.

2. IN DIVISION THE NUMERATOR MUST BE LARGER THAN THE DENOMINATOR IF A QUOTIENT LARGER THAN ZERO IS DESIRED.

   I.E. 13784/100 = 138.

2. BRACKETS AND PRECEDENCE

IN THE PREFACE AN EXAMPLE IS GIVEN TO ILLUSTRATE THE IMPORTANCE OF UNIFORMITY OF ORDER OF EXECUTING OPERATIONS IN A STATEMENT. THIS ORDERING IS FURTHER EXPLAINED SO THAT THE PROGRAMMER CAN CALCULATE THE MANNER IN WHICH THE ACT I SYSTEM WILL SYNTHESIZE EACH STATEMENT AND CREATE THE OBJECT PROGRAM.

RULE 1. THE ACT I SYSTEM WILL WORK FROM THE INNER MOST BRACKETS OUT, I.E., a + [b + c]

b + c
a + [ ]

RULE 2. WITHIN A BRACKET LEVEL, OPERATIONS ARE DONE IN ORDER OF THEIR PRECEDENCE. I.E., [a x sin b + c]

sin b
a x sin b
a x sin b + c

GENERALLY FUNCTIONS WILL BE PRECEDENCE 3, MULTIPLICATION AND DIVISION WILL BE PRECEDENCE 2, ADDITION AND SUBTRACTION WILL BE PRECEDENCE 1, AND SUBSTITUTION WITH BASIC MACHINE LANGUAGE WILL BE PRECEDENCE 0. A TABLE OF CODES BY ASSIGNED PRECEDENCE FOR OPERATIONS USED IN THE MANUAL ARE AS FOLLOWS:

PRECEDENCE 3 - less, grt, equal, abs, print, read, cr, tab, flo, unflo, finp, fprt

PRECEDENCE 2 - x, /, fx, f/

PRECEDENCE 1 - +, -, f+, f-

PRECEDENCE 0 - : when, trn, stop, ret, use, bring, add, subtr, mult, div, extrt, hold, clear, stadd, rdhex.

RULE 3. WITHIN A BRACKET LEVEL AND OPERATIONS OF THE SAME PRECEDENCE, THE OPERATIONS ARE DONE IN ORDER FROM LEFT TO RIGHT. I.E., [a / b x c]

a / b
[a / b] x c

EXAMPLES:

1. a + b : c 0 print c

NOTE THE FIRST OPERATION TO BE PERFORMED WILL BE 0 print c WHICH WILL PRINT OUT WHAT WAS IN c BEFORE a + b IS STORED THERE. THIS MAY BE CORRECTED IN EITHER OF THE FOLLOWING WAYS:

a + b : c
0 print c

[a + b : c] 0 print c

2. a + b grt c + d x f trn sl7

NOTE THE grt IS PRECEDENCE 3 SO IT WILL NOT HAVE ENOUGH INFORMATION TO MAKE THIS TEST UNTIL (a + b) AND (c + dxf) IS CALCULATED. THIS MAY BE CORRECTED IN THE FOLLOWING MANNER:

[a + b] grt [c + dxf] trn sl7

3. SUBROUTINES

A. THE BASIC PACKAGE OF SUBROUTINES THAT IS USED WITH THE ACT I SYSTEM IS:

PROGRAM	STORAGE	LOAD
		;000xxxx
TRACE, FLOAT, UNFLOAT	2 TRACKS	/000xxxx
FLOATING POINT OPERATIONS	6 TRACKS	;000xxxx + 200 /000xxxx + 200
FIXED POINT INPUT-OUTPUT	2 TRACKS	;000xxxx + 800 /000xxxx + 800

THE TAPES OF ACT I THAT ARE DISTRIBUTED HAVE XXXX = 5000. THE SUBROUTINE PACKAGE CAN BE RELOCATED TO OTHER POSITIONS IN MEMORY BY CHANGING THE "START FILLS" AND "SET MODIFIERS". THE PACKAGE MUST BE IN 10 CONSECUTIVE TRACKS AND IN THE SAME ORDER AS ABOVE. IF THE SUBROUTINE PACKAGE IS RE-LOCATED, THE TRANSFER ADDRESSES OF THE OPERATION CODES MUST BE CHANGED.

SUBROUTINE PACKAGE PROGRAM STOPS.

5110 - UNFLOAT - NUMBER OF DECIMAL DIGITS IS TOO LARGE

5303 - FLOATING - DIVISION BY ZERO.

5429 - FLOATING - CHARACTERISTIC IS 32.

B. SUBROUTINES THAT ARE USED WITH STATEMENTS I.E. SUB XYZ A B C

THIS INDICATES THAT ARGUMENT A AND B ARE TO BE USED WITH SUBROUTINE XYZ AND THAT THE RESULT IS TO BE PLACED IN C. THE ACT I COMPILER WILL BUILD THE FOLLOWING SET OF INSTRUCTIONS:

```
B       A
XR      0310
U       (    ) FIRST LOCATION OF REGION XYZ
Z       ( B )
Z       ( C )
```

THE XYZ SUBROUTINE WILL FIND THE FIRST ARGUMENT IN THE ACCUMULATOR. THE OTHER ARGUMENT(S), THE LOCATION TO PLACE THE RESULT AND THE EXIT LOCATION MAY BE CALCULATED FROM LOCATION (0310).

4. Assembly for ACT I

Generate Operation Codes For ACT I

A. Function:

To generate and store all necessary codes, symbols, and instructions for an operation table to be used by the ACT I Compiler.

B. Input: (All in 6-bit mode)

1. For each ACT I operation:

   a. The symbol of the operation.
   b. The precedence (0, 1, 2, or 3).
   c. Symbolic instructions to be coded by ACT I. There will be N of these, and they must be in one of the following formats:

      1. ØlO' - Denotes left operand needed.
      2. ØrO' - Denotes right operand needed.
      3. Øio' - Denotes intermediate operand needed.
      *4. ØcA' - Denotes constant address
      *5. ØjA' - Denotes jump address
      *6. ØtA' - Denotes transfer address

   d. An extra stop code to denote the end of the operation.

   * The 4-digit address must follow the corresponding symbolic instruction.

2. A stop code to indicate the end of the operation table.

3. For ITER and SUB, only the symbols are entered (omit Input B,C, & D)

C. Output:

Hex. punchout of the following parts of memory (with check sums). ACT I must be in LGP-30.

A. In track 36: The 6-bit symbols as read under Input shifted left one bit. A (-1) @ Q of 3 follows the last symbol.

B. In track 37: In sectors corresponding to symbols of track 36, code words in the following bit format.

Bits	Reserved for
s,1-3	Zero
4-5	Precedence (0 -3)
6	Operation indicator
7	Left op. indicator 0 means no 1 means yes
8-11	N-1 (instructions to generate)
12-18	(location of 1st. instruction for ACT I to code -3800) e.g. $(1000000)_2$ is 3900
19-30	Transfer address-location of the transfer to a subroutine.

ITER and SUB are identical with their track 36 counterparts. The above track 37 bit format is ignored.

C. In tracks 38 and 39: The actual instructions to be coded by the ACT I compiler, with bits to the left of the operation bits as follows:

"Op". Required	Sign	7	8	9	10	Hex Instruction
Left op	0	0	0	0	0	Ø 0000
Right op	0	1	0	0	0	10 Ø 0000
Intermed op	0	1	1	0	0	18 Ø 0000
Jump op.	0	1	1	1	0	1J Ø $T_1T_2S_1S_2$
Abs.address	0	1	1	1	1	1Q Ø $T_1T_2S_1S_2$
Transfer op.	1	0	0	0	0	800Ø 0000

Nothing is put in tracks 38 or 39 for ITER or SUB.

ERROR HALTS:

ERROR NUMBER	LOCATION	MEANING
1	Lo + 04	6-BIT BUTTON NOT DEPRESSED
2	120	PRECEDENCE IS INCORRECT
3	17	OP.SYMBOL TABLE IS FULL
4	259	TRACKS 38 AND 39 ARE FULL
5	324	SYMBOLIC INSTRUCTION IS NOT IN CORRECT FORMAT (SEE INPUT PARAGRAPH 4, A THRU F)

PROGRAMMED STOPS	LOCATION	MEANING
	Lo + 40	TURN PUNCH ON
	Lo + 209	FINISHED-DEPRESS START FOR NEXT DATA.

EXAMPLE

THE TAPE OF OPERATIONS CODES CONTAINS THE OPERATIONS (+,cr) AND THE NECESSARY INFORMATION FOR PROGRAM GENERATION.

```
+'l'blo'aro'hio''
cr'3'pca'1600'zca'0'''
```

THESE HAVE INPUT MEANINGS DESCRIBED IN 4.B.1

A,B,C,C,C,D
A,B,C, (ADDRESS), C, (ADDRESS), D

THE OUTPUT IN HEX. IS AS FOLLOWS:

IN TRACK 36	IN TRACK 37	IN TRACK 38	
00000016	07200000	000B0000	FOR +
00001F9F	0210J000	010A0000	
W0000000		018H0000	
		01QP1000	FOR CR
		01QZ0000	

STORAGE: 4 TRACKS OF INSTRUCTIONS AND CONSTANTS. PROGRAM MAY BE RELOCATABLE.

TRACK 63 IS USED FOR TEMP. STORAGE. THE ACT I COMPILER MUST BE IN 4000 THRU 5763 FOR THE PUNCH-OUT PORTION OF THIS PROGRAM.

ROYAL McBEE • *data processing division*

PORT CHESTER, N. Y. • OFFICES COAST-TO-COAST, IN CANADA AND ABROAD