A.1 General Form of an L756 Program

An L756 program is a sequence of (possibly labeled) statements. Normally, statements are executed in their order of appearance in the sequence unless a transfer of control is specified by a GOTO statement, an IF statement, or a FOR statement.

At least one blank must separate adjacent identifiers, integers and/or reserved words, and no blank may separate adjacent characters of a symbol. The input cards have no logical meaning, i.e., they are viewed as one continuous stream of characters.

Example

```
      LET X = 12
      IF X > 10 THEN GOTO L30
L30;  END
```

A.2 Numbers

A number is a sequence of decimal digits such as:

```
1235
22
```

A.3 Variables/Labels

A variable is denoted by a string of letters and digits whose first character is a letter. Strings which agree on their first seven characters are assumed to denote the same variable. Labels are similarly defined but they must be followed by ":".

```
I
I1AB3
Z9:
00:
```

Each variable must be defined at the start of the program using a DCL instruction of the form

```
DCL <id>
```

A.4 Arithmetic Expressions

Arithmetic expressions are formulas for computing a value. They are made up using numbers, variables, and the four arithmetic operators.

```
+ Addition
- Subtraction
· Multiplication
/ Division
```

as well as the left and right parentheses.

Examples of general expressions:

```
A1 + B * (C + D (C + E))
A1 + 3114159/4*(C + 12)
```

The order in which operations are performed is normally from left to right except that

1. Operations within parentheses are carried out before the parenthesized quantity is use din further computations.
2. Multiplications and divisions are done before additions or subtractions.

Example

The order of operations is indicated below the following expression:

```
A + (B/(C + D)) * (F * G * (H + B)) + C
  7   2   1     6    3   5    4     8
```

A.5 Statements

The following are the allowable types of statements.

1. Assignment Statement

   The general form of an Assignment statement is

   ```
   LET <variable> = <arithmetic expression>
   ```

   The value of the arithmetic expression is computed and assigned to the variable.

   Example

   ```
   LET X1 = Y1 + 1206 * Z + XL
   ```

2. GOTO Statement

   The general form of a GOTO statement is

   ```
   GOTO <label>
   ```

   Control is transferred to the statement with the specified label.

   Example

   ```
   GOTO L5
   ```

3. Conditional Statement

   The general form of a Conditional Statement is:

```
IF <arithmetic expression><relational operator>
<arithmetic expression> THEN GOTO <label>
```

where the relational operators are

```
=    equal
<>   not equal
<    less than
<=   less than or equal
>    greater than
>=   greater than or equal
```

If the relation holds between the values of the two arithmetic expressions, control is transferred to the specified label. Otherwise, control passes to the ?.

Examples

```
IF X > Y THEN GOTO L2
IF Z + (X*Y) = X1 + 12 i THEN GOTO L5
```

4. FOR Statement

FOR statements are used to set up program loops. There are two general forms of FOR Statements:

```
FOR <variable> = <arithmetic expression 1>
STEP <arithmetic expression 3> TO
<arithmetic expression 2>

DO ... statements ... NEXT

FOR <variable> = <arithmetic expression 1> TO
<arithmetic expression 2>  DO ... statements ... NEXT
```

The second form is interpreted in the same way as the first except that all the step size (arithmetic expression 3) is assumed equal to 1. The FOR statement forms the entrance of a program loop that continues through any number of subsequent statements until a NEXT to close the loop is reached.

Example

```
FOR X = 1 STEP 5 TO 100 DO
   LET W = X + Y + Z
   LET Z = X * Y
NEXT
```

Informally, the statements in the loop between the FOR and the NEXT

are executed over and over again a number of times as specified by the FOR statement. Each time around the loop the value of the variable is incremented by the step and compared with the final value to determine whether to enter the loop again or to terminate the FOR statement.

Specifically, the execution of the FOR Statement is as follows:

a. Arithmetic expression 1 (the **initial value**), arithmetic expression 2 (the **final value**), and arithmetic expression 3 (the **step size**) are all computed.

b. The **initial value** is assigned to the variable.

c. A test is made, depending on the value of the **step size**.

- If the **step size** is greater than or equal to zero, and the value of the variable is less than or equal to the **final value**, control passes to the statement following the FOR statement (i.e., the loop is entered). Otherwise, control passes to the statement following the associated NEXT (i.e., the loop is not entered and the FOR statement is completed).

- If the **step size** is negative, and the value of the variable is greater than or equal to the **final value**, control passes to the statement following the FOR statement (i.e., the loop is entered). Otherwise, control passes to the statement following the associated NEXT (i.e., the loop is not entered and the FOR statement is completed).

- If the loop is entered, the statements between the FOR statement and the associated NEXT are executed.

- When NEXT is reached, the value of the **step size** is added to the current value of the variable and the result assigned to the variable. The procedure then continues at step c). Note the **initial value**, **final value**, and **step size** are computed only once, at the beginning, not every time around the loop. However, the value of the controlled variable can be changed by the statements within the loop and the latest value is always used for the incrementing and testing steps.

  It is allowable to have loops within loops, but they must be properly nested.

  Example

```
FOR X = 1 TO N
...
FOR Y = 1 TO N
...
```

```
NEXT
...
NEXT
```

5. input/output statements

```
READ  <identifier>
WRITE <expression>
```

6. END Statement

Every program must have a single END statement -- the last (highest-numbered) statement in the program. The general form of an END statement is

```
END
```

7. REM Statement

Comments can be added to a program using the REM statement, the general form of which is

```
REM <any sequence of characters but;>;
```

The compiler ignores the input stream between (and including) the REM and its first following semicolon.