TRT D1(L1,B1),D2(B2)

Translate and Test

SS_1

Op Code	LL_1	B_1D_1	D_1D_1	B_2D_2	D_2D_2

TRT is used to scan a string of characters, searching for the occurrence of any the characters which are specified in a translate table. The translate table is usually 256 bytes in length - one for each possible EBCDIC character that might be encountered during the search. Unlike **TR** which provides automatic translation of character strings, **TRT** sets up the conditions for a translate to occur, but leaves the programmer with the task of making the translation. In many cases, **TRT** is used for scanning strings with no intention of making a translation.

Operand 1 designates a string contained in memory which is to be scanned and possibly translated. Operand 2 designates the translate table. This table is also called a “table of functions”. **TRT** scans one byte at a time, starting with the leftmost byte of Operand 1, proceding from left to right. Each “string byte” that is scanned is used as a displacement into the table. The byte in the translate table at the given offset is called the “function byte”. If the function byte is X’00’, the scanning process continues with the next byte in Operand 1. In other words, a X’00’ in a function byte indicates that we are not interested in finding the string byte that was used as a displacement. On the other hand, if the function byte is not X’00’, then **TRT** modifies register 1 to contain the address of the current string byte and also modifies register 2 to contain the corresponding function byte in bits 24 - 31 (the rightmost byte of the register). The **TRT** operation is terminated when a nonzero function byte is encountered. At this point the programmer is free to make the translation using registers 1 and 2.

Since a string byte can contain any value from X’00’ to X’FF’, the table of functions is usually 256 bytes long to accommodate the range of addresses from table + X’00’ to table + X’FF’. Let us consider how a particular string byte is processed using a table of functions called “TABLE”. For example, how would a byte containing X’A2’ be processed? Since X’A2’ = 162 in base 10, the function byte at address “TABLE+162” would be examined. If the function byte were X’00’, execution would continue with the next string byte (moving from left to right within the string). If the function byte contained something different from X’00’, then register 1 would contain the address of the current string byte and register 2 would contain the corresponding function byte.

TRT also sets the condition code to indicate the results of the scanning operation as follows:

Condition Code	**Indications**
0 (Zero)	All function bytes encountered were X’00’.
1 (Minus)	A nonzero function byte was found before the end of operand 1
2 (Positive)	A nonzero function byte was found at the end of the operand 1

The condition code can be tested using BZ, BNZ, BM, BNM, BP, or BNP.

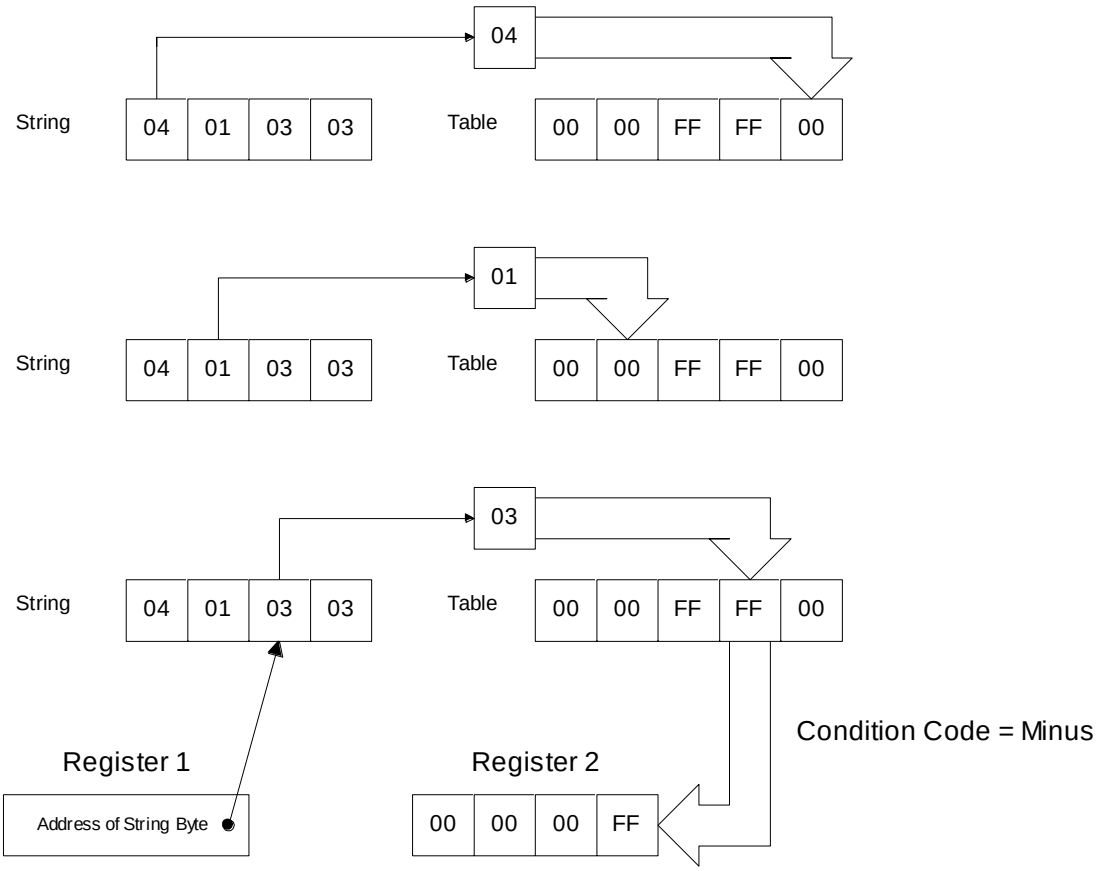
Consider the following example. To keep things simple, we begin with a small (5 bytes) table and assume that the bytes in the string which is being scanned generate displacements that stay inside the table. We assume the following definitions.

```
STRING    DC   X'04010303'
TABLE     DC   X'0000FFFF00'
```

Notice that the table indicates that we are interested in finding a string byte that contains X'02' or X'03'. Those bytes contain X'FF'. We are not interested in scanning for X'00', X'01', or X'04' since those table bytes contain X'00'.

We issue the following command.

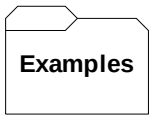
```
TR   STRING,TABLE
```



The previous diagram indicates the result of the **TRT**. First the string byte X'04' is used as a displacement into the table of functions. The function byte in this position is X'00'. As a result, execution continues with the next string byte. Using X'01' as a displacement we find the function byte X'00'. Execution continues with the next string byte. Finally, X'03' is used as a displacement into the table of functions and we find a nonzero function byte X'FF'. Execution of TRT terminates at this point with the condition code set to "Minus" to indicate that a nonzero function byte was found before the end of the string.

When defining a TRT table we are usually interested in finding only a few selected characters. This means we can begin defining a TRT table by starting with a table of all zeros and use a collection of ORG's to redefine the pertinent characters. The following example shows how to code a table to scan for a dollar sign or a question mark.

```
TABLE     DC    256AL1(0)
ORG   TABLE+C'$'   Scan for a dollar sign
DC    X'FF'
ORG   TABLE+C'?'   Scan for a question mark
DC    X'FF'
ORG
```



Some Unrelated TRT's:

```
TABLE    DC    256AL1(0)
          ORG      TABLE+C'*'
          DC       C'X'        ANY NONZERO VALUE IS VALID
          ORG      TABLE+C'S'
          DC       C'Y'        ANY NONZERO VALUE IS VALID
          ORG
STRING1   DC    X'DOGS*****'
STRING2   DC    X'*SSS'
STRING3   DC    X'CATS'
STRING4   DC    X'ABCDEFGHIJKLM'
                ...             Result:
          TR    STRING1,TABLE   R1 POINTS AT "S" IN STRING1
                                R2 POINTS AT "Y" IN TABLE
                                CONDITION CODE = MINUS
          TR    STRING2,TABLE   R1 POINTS AT "*" IN STRING1
                                R2 POINTS AT "X" IN TABLE
                                CONDITION CODE = MINUS
          TR    STRING3,TABLE   R1 POINTS AT "S" IN STRING1
                                R2 POINTS AT "Y" IN TABLE
                                CONDITION CODE = POSITIVE
          TR    STRING4,TABLE   R1 AND R2 ARE UNCHANGED
                                CONDITION CODE = ZERO
```

Tips

1. **TRT** can be used to test a field for numeric data (X'F0' - X'F9') by using the following table.

```
TABLE       DC    256X'FF'
            ORG   TABLE+X'F0'
            DC    10X'00'     10 DIGITS OCCUR IN ORDER
            ORG
```

Suppose we want to test a field called "FIELD" to see if it is numeric in the sense described above. This can be accomplished as follows.

```
            TRT   FIELD,TABLE
            BZ    ALLNUMS
```

If all the bytes in "FIELD" are numeric, they will be translated to table bytes that contain zeros and the condition code is set to zero. If a nonzero byte is found, the condition code is either minus or positive and the branch is not taken.

This technique can be used to verify that a field contains only characters from a given subset.