CLI is used to compare two fields that are both in storage. Operand 1 is a field in main storage, while the second operand is a self-defining term that gets assembled as a one byte immediate constant (II2) in the second byte of the object code of the CLI instruction. Only the first byte of Operand 1 is compared to the immediate constant. The comparison is made based on the ordering of characters in the EBCDIC encoding.

Executing a compare instruction sets the condition code (a two bit field in the PSW) to indicate how operand 1 (target field) compares with operand 2 (immediate constant). The condition code is set as follows,

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Condition Code Value</th>
<th>Test With</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operand 1 equals Operand 2</td>
<td>0 (equal)</td>
<td>BE (Branch Equal)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BNE (Branch Not equal)</td>
</tr>
<tr>
<td>Operand 1 is less than Operand 2</td>
<td>1 (low)</td>
<td>BL (Branch Low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BNL (Branch Not Low)</td>
</tr>
<tr>
<td>Operand 1 is greater than Operand 2</td>
<td>2 (high)</td>
<td>BH (Branch High)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BNH (Branch Not High)</td>
</tr>
</tbody>
</table>

The table above also illustrates the appropriate branch instructions for testing the condition code. When comparing two fields, a CLI instruction should be followed immediately by one or more branch instructions for testing the contents of the condition code:

```
FIELD DC C’1234’
... CLI FIELD, C’A’
BH AHIGH BRANCH IF FIELD IS HIGH
BL BHIGH BRANCH IF FIELD IS LOW
```

In the example above, the first byte of FIELD, which contains the character “1” and is represented as x’F1’, is compared to the self-defining term C’A’, which assembles as a x’C1’. In EBCDIC, since x’F1’ is greater than x’C1’, the condition code is set to “high” to indicate that operand 1 is “higher” than operand 2.

The following example illustrates how a CLI might be processed by the assembler.

```
LOC OBJECT CODE
000F12 95F4C044 CLI CUSTCODE, C’4’
... 001028 CUSTCODE DS CL1
```
In the example above, the op-code for CLI is x’95’, the self-defining term C’4’ is assembled as the one byte hexadecimal constant x’F4’, and CUSTCODE is translated into the base/displacement address C044.

**Examples**

Some Unrelated CLI’s:

<table>
<thead>
<tr>
<th>J</th>
<th>DC</th>
<th>C’ABC’</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>DC</td>
<td>C’DEF’</td>
</tr>
<tr>
<td>L</td>
<td>DC</td>
<td>C’GH’</td>
</tr>
<tr>
<td>M</td>
<td>DC</td>
<td>C’12345’</td>
</tr>
</tbody>
</table>

Result:

CLI J, C’A’  Condition Code = Equal, one byte compared.
CLI J, C’B’  Condition Code = Low.
CLI J, C’5’  Condition Code = Low, letters < numbers.
CLI K, X’C4’ Condition Code = Equal.
CLI L, C’A’  Condition Code = High.
CLI L, =C’G’ Assembly error, Literals not allowed.
CLI B, X’C1C2’ Assembly error, 1-byte comparisons only.
CLI C’A’, M Assembly error, operands out of order.
CLI A(20), B Assembly Error, 1-byte comparisons only.
CLI A, B(20) Assembly Error - operand 1 determines the length

**Tips**

1. Use CLI instead of CLC when comparing 1-byte fields. The resulting code is smaller and slightly more efficient. More importantly, it makes explicit the fact that you are comparing two 1-byte fields.