CLI

Compare Logical Immediate

SI

Op Code	II_2	B_1D_1	D_1D_1

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 (II_2) 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,

Comparison	Condition Code Value	Test With
Operand 1 equals Operand 2	0 (equal)	BE (Branch Equal) BNE (Branch Not equal)
Operand 1 is less than Operand 2	1 (low)	BL (Branch Low) BNL (Branch Not Low)
Operand 1 is greater than Operand 2	2 (high)	BH (Branch High) BNH (Branch Not High)

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:

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

In the example above, the first byte of FIELDA, 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:

```
J        DC    C’ABC’
K        DC    C’DEF’
L        DC    C’GH’
M        DC    C’12345’
                       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.
         CLC   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.