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

Compare Logical Characters

SS_1

Op Code	LL_1	B_1D_1	D_1D_1	B_2D_2	D_2D_2

CLC is used to compare two fields that are both in storage. The fields are compared, one byte at a time beginning with the bytes specified in addresses $B_1D_1D_1D_1$ and $B_2D_2D_2D_2$, and moving to higher addresses in the source and target fields. Each byte in the source is compared to a byte in the target according to the ordering specified in the EBCDIC encoding sequence. 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 (source field). 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 indicates the appropriate branch instructions for testing the condition code. When comparing two fields, a **CLC** instruction should be followed immediately by one or more branch instructions for testing the contents of the condition code:

```
CLC      FIELDA,FIELDB
BH       AHIGH         BRANCH IF FIELDA IS HIGH
BL       BHIGH         BRANCH IF FIELDA IS LOW
```

Bytes are compared until the number of bytes specified (implicitly or explicitly) in operand 1 have been exhausted or until two unequal bytes are found - whichever occurs first. As you can see from the instruction format above, the instruction carries with it the maximum number of bytes to be compared, as well as the beginning addresses of the source and target fields. Notice that the instruction does not specify the ending addresses of either field - the instruction is no respecter of fields. If a longer field is compared to a shorter field, the bytes following the shorter field may be used in the comparison operation.

The length (LL_1) determines the maximum number of bytes which will be compared. The length is usually determined implicitly from the length of operand 1 but the programmer can provide an explicit length. Consider the two example CLC's below,

```
Object code               Assembler code

               FIELDA    DC    CL4'ABCD'
               FIELDB    DC    C'ABE'
                         ...
D502C00CC008             CLC   FIELDB,FIELDA     Implicit length = 3,
                                                 COND CODE = HIGH
D501C00CC008             CLC   FIELDB(2),FIELDA  Explicit length = 2,
                                                 COND CODE = EQUAL
```

In the first **CLC**, 'A' in FIELDB is compared with 'A' in FIELDA, then 'B' in FIELDB is compared with 'B' in FIELDA, finally, 'E' in FIELDB is compared with 'C' in FIELDA. At this point, the condition code is set to 'HIGH' since 'E' follows 'C' in the EBCDIC encoding sequence. In the second example, 'A' in FIELDB is compared with 'A' in FIELDA, then 'B' in FIELDB is compared with 'B' in FIELDA. The condition code is set to 'EQUAL' since an explicit length of 2 was coded.

Examples

Some Unrelated CLC's:

```
   A        DC    C'PQR'
   B        DC    C'ABCD'
   C        DC    C'PQ'
   D        DC    P'12'       D = X'012C'
            ...         Result:
            CLC   A,B     Condition Code = High, one byte compared.
            CLC   A(2),C  Condition Code = Equal, two bytes compared.
            CLC   C,A     Condition Code = Equal, two bytes compared.
            CLC   A,=C' ' Condition Code = High, one byte compared.
This
                          coding is unwise since it sets up the
                          possibility that bytes following the blank
                          literal in the literal pool might become part
                          of the comparison.
            CLC   A,=CL3' '  Condition Code = High, this is a better
                          version of the previous comparison.  Should
                          length of "A" change, an error may occur.
            CLC   B,=X'C1C2C3C4'  Condition Code = Equal,
                           4 bytes were compared.
            CLC   D,=P'12' This is a dangerous compare since the data
                           involved is in packed format.  If the
                           fields are unchanged from their assembly
                           time format the condition code would be
                           equal.
            CLC   A(500),B  Assembly Error -  max  length is 256
            CLC   A,B(20)   Assembly Error -  operand 1 determines the
                            length
```

Tips

1. As with any storage to storage instruction, you must pay careful attention to lengths of the two operands. Generally, you should be comparing fields that are the same size.

2. The instruction was designed to compare fields that are in character format. It can be used to compare fields with non-character data, but this takes special consideration to make sure the comparison will produce the desired results. Packed decimal data and binary data are supported with their own special comparison instructions.

3. The condition code can be changed by any other type of comparison instruction as well as by a variety of arithmetic instructions. Don’t rely on the condition code to remain set - after you have issued a **CLC** , you should follow it up immediately with a branch instruction.