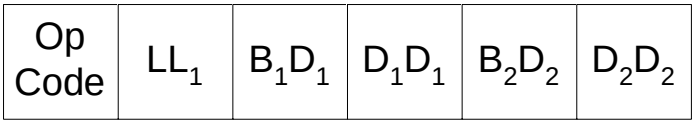
MVC D1(L1,B1),D2(B2)

Move Characters

SS_1



MVC is an instruction which is designed to copy a collection of consecutive bytes from one storage location to another. As you can see from the instruction format above, the instruction carries with it the number of bytes to be copied, 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. **MVC** copies $LL_1 + 1$ consecutive bytes from the storage location designated by $B_2D_2D_2D_2$ to the storage location designated by $B_1D_1D_1D_1$. The copying occurs one byte at a time from Operand 2 to Operand 1, and within each operand, from lower to higher numbered addresses.

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

```
Object code              Assembler code

                FIELDA   DS       CL8
                FIELDB   DS       CL5
                         ...
D207C008C010             MVC      FIELDA,FIELDB     Implicit length
D202C008C010             MVC      FIELDA(3),FIELDB  Explicit length
```

In the first example, the length implicitly defaults to 8, the length of FIELDA. In the second example, the length is explicitly 3. Notice that the assembled length (LL_1) is one less than the implicit or explicit length. This can be seen in the object code above where the assembled lengths are x'07' and x'02'.

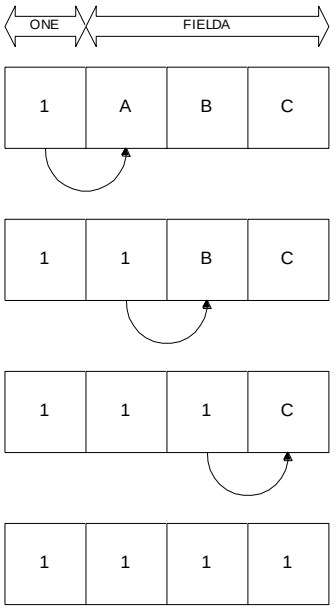
The copying operation is usually straightforward, but can be more complicated by overlapping the source and target fields. Keep in mind that the copy is made one byte at a time. Consider the following examples,

```
Object code              Assembler code

                ONE      DC    C'1'
                FIELDA   DC    CL3'ABC'
                FIELDB   DC    CL3'DEF'
                FIELDC   DC    CL4'1234'
                         ...
D202C008C00B             MVC   FIELDA,FIELDB    After FIELDA = 'DEF'
D201C00EC008             MVC   FIELDC,FIELDA    After FIELDC = 'ABCD'
D201C008C007             MVC   FIELDA,ONE       After FIELDA = '1111'
```

In the first **MVC** above, 3 consecutive bytes in FIELDB are simply copied to FIELDA. In the second example, 4 consecutive bytes are copied into FIELDC (implicit length = 4) from FIELDA. Since FIELDA was only 3 bytes long the fourth byte was copied from the first byte of the next field - FIELDB. The third **MVC** is complicated by the fact that the source and target fields overlap. We will examine the third move in some detail.

```
                    MVC    FIELDA,ONE    THIS IS A 3 BYTE MOVE
```



First byte of source copied to first byte of target.

Second byte of source copied to second byte of target.

Third byte of source copied to third byte of target.

The example above depends heavily on the fact that the source and target fields overlap and that bytes are copied one at a time. In fact, it is common to use this technique to clear fields. Assume you have a buffer you would like to clear to spaces. By defining a single blank directly in front of the field you want to clear and moving the blank field to the buffer, the blank can be propagated throughout the buffer:

```
          MVC   BUFFER,BLANK          BLANK IS PROPAGATED
          ...
BLANK     DC    C''       BLANK MUST IMMEDIATELY PRECEDE…
BUFFER    DS    CL133     …THE BUFFER
```

Examples

Some Unrelated MVC's:

```
A         DC    C'123'
B         DC    C'ABCD'
C         DC    C'PQ'
          ...              Result:
          MVC   A,B        A = 'ABC'     B = 'ABCD'
          MVC   A+1,B      A = '1AB'     B = 'CBCD'
          MVC   A+1(2),B   A = '1AB'     B = 'ABCD'
          MVC   B,=C'XY'   B = 'XY??'    Two bytes copied from
```

```
                                          the literal pool, two
                                          unknown bytes are copied
            MVC   B,B+1          B = 'BCDP'   Left shift
            MVC   B+1,B          B = 'AAAA'   First byte is propagated
            MVC   C,A            C = '12'  A = '123'  Truncation
            MVC   A(L'C),C       A = 'PQ3'   Explicit Length attribute
            MVC   A(1000),B    Assembly Error - max length is 256
bytes
            MVC   A,B(20)      Assembly Error - Op-1 determines length
```

Tips

1. Pay attention to the lengths of the fields involved in any MVC statement. If the target field is longer than the source field, bytes following the source may be transferred. If the target field is shorter than the source field, bytes from the source may may be truncated.

2. Be careful of using literals in an MVC, since stray bytes in the literal pool will be moved if the specified length is longer than the literal. Lengths can also be specified in a literal (=CL133' '). The following typical error MVC BUFFER,=C' ' can be fixed as MVC BUFFER,=CL80' '.