**MVZ** performs in a way that is analogous to **MVC**. While MVC works on entire bytes, MVZ only processes the zoned parts (leftmost 4 bits) of the bytes it references. The purpose of a move zones instruction is to move the zoned parts of a consecutive collection of bytes from one location in memory to another location. As you can see from the instruction format above, the instruction carries with it the number of bytes to be copied (LL₁), as well as the beginning addresses of the source (B₁D₁D₁D₁) and target (B₂D₂D₂D₂) fields. Notice that the instruction does not specify the ending addresses of either field - the instruction is no respecter of fields. MVZ copies the zoned parts of LL₁ + 1 consecutive bytes from the storage location designated by B₂D₂D₂D₂ to the storage location designated by B₁D₁D₁D₁.

The length (LL₁) determines the number of “half-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 MVZ’s below.

<table>
<thead>
<tr>
<th>Object code</th>
<th>Assembler code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIELDA    DS        CL8</td>
<td></td>
</tr>
<tr>
<td>FIELDB    DS        CL5</td>
<td></td>
</tr>
<tr>
<td>D307C008C010         MVZ    FIELDA,FIELDB Implicit length</td>
<td></td>
</tr>
<tr>
<td>D302C008C010         MVZ    FIELDA(3),FIELDB Explicit length</td>
<td></td>
</tr>
</tbody>
</table>

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 (LL₁) 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.

<table>
<thead>
<tr>
<th>Object code</th>
<th>Assembler code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONE      DC    C’1’</td>
<td>ONE = X’F1’</td>
</tr>
<tr>
<td>FIELDA   DC    CL3’ABC’</td>
<td>FIELDA = X’C1C2C3’</td>
</tr>
<tr>
<td>FIELDB   DC    XL3’123456’</td>
<td>FIELDB = X’123456’</td>
</tr>
<tr>
<td>D302C008C00B         MVZ    FIELDA,FIELDB After FIELDA = X’113253’</td>
<td></td>
</tr>
<tr>
<td>D302C008C008         MVZ    FIELDB,FIELDA After FIELDB = X’C2C4C6’</td>
<td></td>
</tr>
<tr>
<td>D302C008C007         MVZ    FIELDA,ONE After FIELDA = X’F1F2F3’</td>
<td></td>
</tr>
</tbody>
</table>
In the first $\text{MVZ}$ above, 3 consecutive zoned half-bytes in $\text{FIELDB}$ are simply copied to the zoned portions of $\text{FIELDA}$. The half-bytes are copied, one at a time moving left to right within both operands. In the second example, 3 consecutive bytes are copied into $\text{FIELDB}$ (implicit length = 3) from $\text{FIELDA}$. The third $\text{MVZ}$ is complicated by the fact that the source and target fields overlap. We will examine the third move in some detail.

\[ \text{MVZ} \quad \text{FIELDA},\text{ONE} \quad \text{THIS IS A 3 BYTE MOVE} \]

\begin{align*}
\text{F1} & \quad \text{C1} & \quad \text{C2} & \quad \text{C3} \\
\text{F1} & \quad \text{F1} & \quad \text{C2} & \quad \text{C3} \\
\text{F1} & \quad \text{F1} & \quad \text{F2} & \quad \text{C3}
\end{align*}

First half-byte of source copied to first half-byte of target.

Second half-byte of source copied to second half-byte of target.

Third half-byte of source copied to third half-byte of target.

---

**Examples**

Some Unrelated $\text{MVZ}$'s:

\begin{align*}
\text{A} & \quad \text{DC} \quad \text{X'123456'} \\
\text{B} & \quad \text{DC} \quad \text{X'ABCDEF'} \\
\text{C} & \quad \text{DC} \quad \text{X'A1B2'} \\
\ldots & \\
\text{MVZ} & \quad \text{A},\text{B} \quad \text{A} = \text{X'A2C4E6'} \quad \text{B} = \text{X'ABCDEF'} \\
\text{MVZ} & \quad \text{A+1},\text{B} \quad \text{A} = \text{X'12A4C6'} \quad \text{B} = \text{X'EBCDEF'} \\
\text{MVZ} & \quad \text{A+1(2)},\text{B} \quad \text{A} = \text{X'12A4C6'} \quad \text{B} = \text{X'ABCDEF'} \\
\text{MVZ} & \quad \text{B},=\text{X'D1E2'} \quad \text{B} = \text{X'DBED?F'} \quad \text{One byte copied from} \\
 & \quad \text{the literal pool} \\
\text{MVZ} & \quad \text{B},\text{B+1} \quad \text{B} = \text{'CBEADAP'} \quad \text{Left shift} \\
\text{MVZ} & \quad \text{B+1(2)},\text{B} \quad \text{B} = \text{'ABADAF'} \quad \text{1st byte is propagated} \\
\text{MVZ} & \quad \text{C},\text{A} \quad \text{C} = \text{'1132'} \quad \text{A} = \text{X'123456'} \\
\text{MVZ} & \quad \text{A(L'C)},\text{C} \quad \text{A} = \text{'A2B456'} \quad \text{Explicit Length} \\
\text{attribute} & \\
\text{bytes} & \\
\text{MVZ} & \quad \text{A(1000)},\text{B} \quad \text{Assembly Error - max length is 256} \\
\text{MVZ} & \quad \text{A,B(20)} \quad \text{Assembly Error - Op-1 determines length}
\end{align*}
**Tips**

1. Pay attention to the lengths of the fields involved in any **MVZ** 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 in the source may be truncated.