MVI is used to move a one byte immediate constant to a field in storage. Operand 1 denotes the field in main storage, while the second operand is coded as a **self-defining term** that gets assembled as a one byte immediate constant ($II_2$) in the second byte of the object code. Only the first byte of Operand 1 is affected by the move.

As an example, consider the following code,

```
MVI FIELDA, X'C1'
...
FIELDA DC X'123456'
```

After execution, FIELDA contains $X'C13456'$. Only the first byte of the field is altered by the immediate instruction.

The following example illustrates how an **MVI** instruction might be processed by the assembler.

```
LOC         OBJECT CODE
000F12      92F4C044          MVI CUSTCODE, C'4'
...
001028      CUSTCODE   DS CL1
```

In the example above, the op-code for **MVI** is $x'92'$, 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 MVI’s:

```
J DC C’ABC’
MVI J, C’X’  J = C’XBC’
MVI J, C’B’  J = C’BBC’
MVI J, C’5’  J = C’5BC’
MVI J, X’F5’ J = C’5BC’
MVI J, 197   J = C’5BC’
MVI J, =C’5’ ASSEMBLY ERROR - OPERAND 2 NOT A SELF-
```
MVI J(1),X'C5' ASSEMBLY ERROR - LENGTH SPECIFICATION NOT ALLOWED IN OPERAND 1