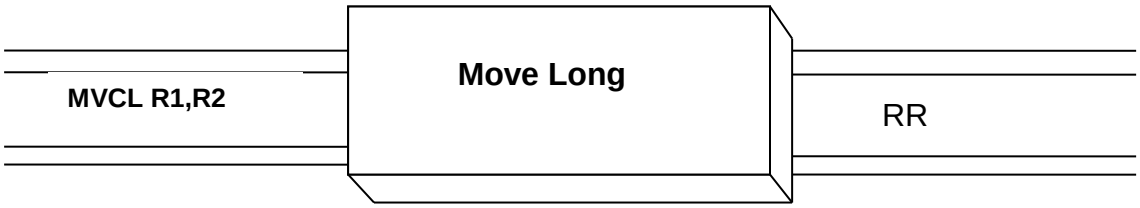


There is a 256 byte limitation when moving character data with MVC. If a field is larger than 256 bytes, one solution to moving it in memory is to divide it into multiple parts and code multiple MVC's. A more elegant solution involves the Move (Characters) Long instruction (MVCL) which was designed for fields longer than 256 bytes. In fact, up to 2^{24} = 16,777,216 bytes can be moved with one MVCL instruction. This instruction is unusual because it requires the use of four registers. Operand 1 and operand 2, denoted by R1 and R2, each represent the even register of an even-odd consecutive register pair (2-3, 4-5, 6-7, …). For example, if you coded "MVCL R4,R8" the registers involved would be R4, R5, R8, and R9. The even registers contain the address of target and source fields, while the odd registers contain the lengths of target and source fields. Additionally, a pad character can be placed in the high order byte of the odd register which contains the length of the source field. The pad character is used to fill up the target field when the source field is shorter and all of its contents have been transmitted. Here is a simple example in which we want to move FIELDB to FIELDA:

```
                            LA    R4,FIELDA           POINT AT TARGET
FIELD WITH EVEN REG
                            L     R5,LENGTHA          PUT LENGTH OF
TARGET IN ODD REG
                            LA    R6,FIELDB           POINT AT SOURCE
FIELD WITH EVEN REG
                            L     R7,LENGTHB          PUT LENGTH OF
SOURCE IN ODD REG
                            ICM   R7,B'1000',BLANK    INSERT A SINGLE
BLANK PAD CHAR IN ODD REG
                            MVCL  R4,R6
                            …
                   FIELDA   DC    CL2000' '
                   BDATA    DC    1000CL1'X'
                            ORG   BDATA
                   FIELDB   DS    CL1000
                   LENGTHA  DC    A(L'FIELDA)         CREATE AN ADDRESS
CONSTANT THAT IS A LENGTH
                   LENGTHB  DC    A(L'FIELDB)         CREATE AN ADDRESS
CONSTANT THAT IS A LENGTH
                   BLANK    DC    C' '
```

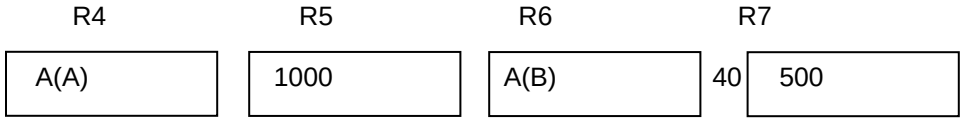
The above code copies 1000 X's from FIELDB into FIELDA. Since FIELDA was larger than FIELDB, the last 1000 bytes of FIELDA are filled with the blank pad character. If the length of FIELDA were equal to or smaller than the length of FIELDB, the pad character would not be used.

Bytes are transferred one at a time in a fashion similar to MVC, by MVCL. As each byte is copied from source to target, the even registers are incremented by one and the odd registers are decremented by one until one of the lengths in the odd registers becomes zero.

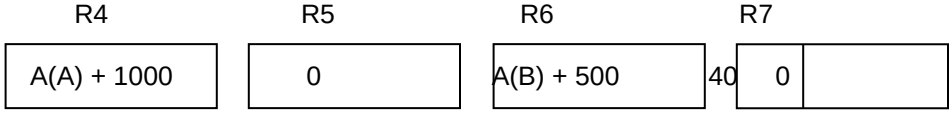
Consider the following scenarios with MVCL in which we are copying from field A to field B. Assume the length of A is L1 and the length of B is L2 . Further assume A(A) and A(B) are the addresses of A and B. Lengths are specified in decimal and we assume a blank pad character in R7.

Case 1: L1 > L2

Before execution:

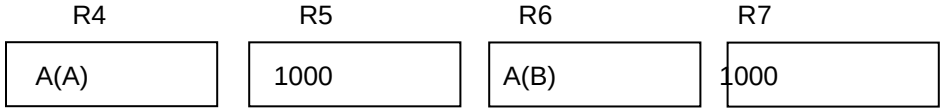


After execution:

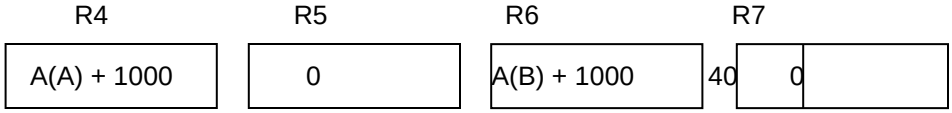


Case 2: L1 = L2

Before execution:

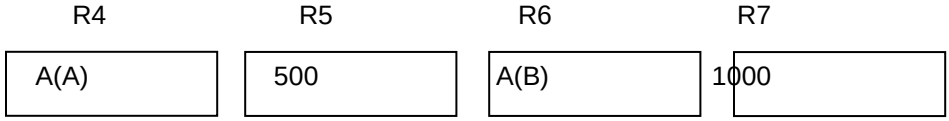


After execution:

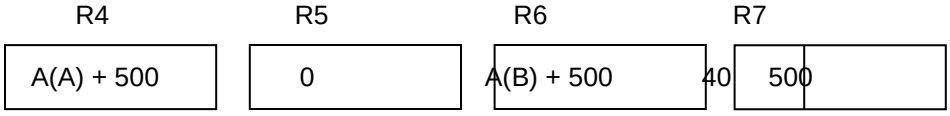


Case 3: L1 < L2

Before execution:

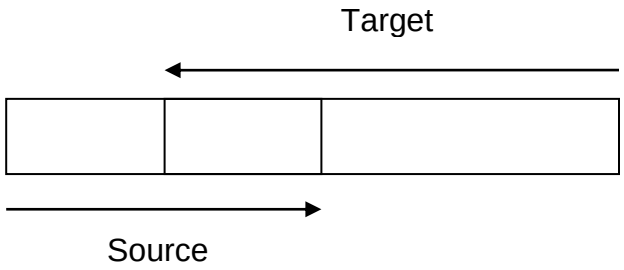


After execution:



The pad byte is only used in case 1 where 500 blank characters are added to FIELDB. In case 2, all 1000 characters of FIELDB are copied to FIELDA. In case 3, only the first 500 bytes of FEILDA are copied to FIELDB.

A destructive overlap occurs when the target and source fields overlap and the address of the source is lower than the address of the target. This is illustrated below.



MVCL detects destructive overlap conditions and prevents any bytes from being moved. Instead, MVCL sets the condition code to indicate destructive overlap. This condition can be tested with BO.

MVCL sets the condition code to 0 (equal) if L1 = L2, to 1 (low) if L1 < L2, to 2 (high) if L1 > L2, and 3 (overflow) if no bytes are moved due to destructive overlap.

Examples

You can use MVCL to blank out a target field by setting the length of the source to 0. This forces the pad character to be copied to each byte of the source:

```
                        LA   R8,TARGET
                        L    R9,TARLEN
                        LA   R4,SOURCE          SOURCE DOESN'T
PARTICIPATE
                        LA   R5,0               SET THE LENGTH OF
SOURCE TO 0
                        ICM  R5,B'1000',BLANK   SET PAD BY TO A BLANK
                        MVCL R8,R4              COPY BLANKS
```

Tips

- It is often difficult to find four available registers when coding MVCL. A simple solution to this problem is to store four consecutive registers in a save area before using MVCL and restore them afterwards:

```
                        STM   R4,R7,SAVE4
                        (use MVCL here…)
                        LM    R4,R7,SAVE4
                        …
            SAVE4       DS    4F                FOUR FULLWORDS FOR
R4,R5,R6, AND R7
```

- Use length attributes for the lengths of the source and target fields:

```
          LENGTHA  DC    A(L'FIELDA)     TURN THE LENGTH INTO A
FULLWORD "ADDRESS"
```