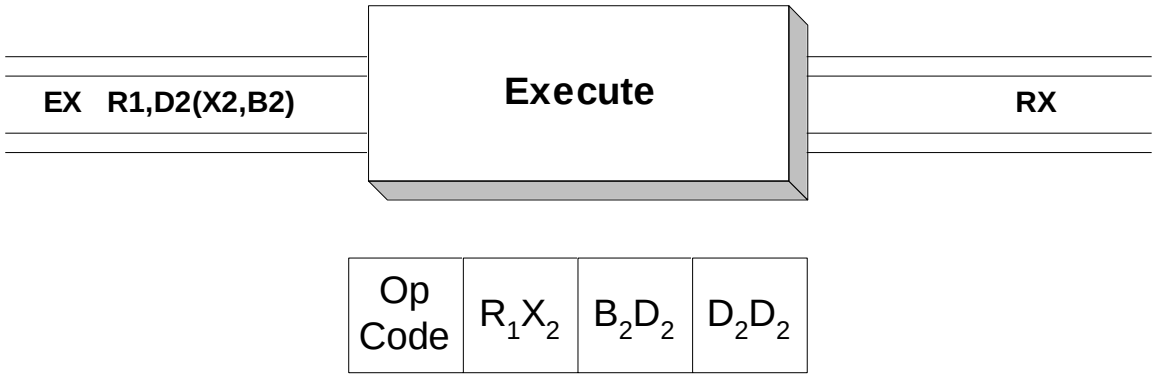
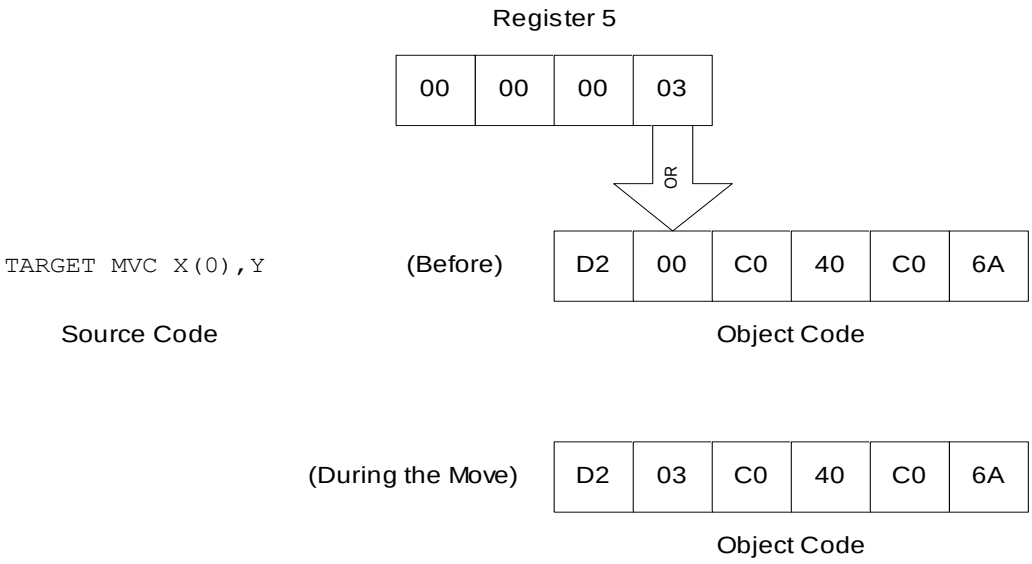


The Execute instruction causes the computer to execute a “target” instruction which is referenced in Operand 2 and which is typically placed out of the normal sequence of instructions, usually among a collection of **DC**’s or **DS**’s. After the target instruction is executed, control returns to the instruction following the **EX** instruction unless the target instruction was a branch. If the target instruction is a conditional branch, then control would resume at the address specified in the branch if the condition being tested is true. If the target instruction is an unconditional branch, then execution would resume at the address specified in the branch instruction. Typically, the target instruction is an **MVC**.

If Operand 1 is not register zero, then the target instruction is temporarily modified before it is executed: The rightmost byte of the register specified by Operand 1 (bits 24 - 31) is “OR-ed” into the second byte of the target instruction. Usually the second byte contains x’00’ which means that the rightmost byte of Operand 1 is copied into the second byte of the target instruction just before the target instruction is executed. In the case where the target is an **MVC** instruction, the second byte is the length byte (the number of bytes to be moved). This means the length can be dynamically changed before executing the **MVC** instruction. This is an important reason to use **EX**.

If Operand 1 is register zero, then the target instruction is executed without modification. Here is an example. We execute the following instruction

```
EX    R5,TARGET
```



In the previous example the target instruction moves a field called “Y” to a field called “X”. Notice that the length in operand 1 is explicitly zero. By coding this we guarantee that the second byte that is assembled for the **MVC** instruction contains x’00’. Before executing the target instruction, the length contained in the rightmost byte of register 5 (x’03’) is “Or-ed” into the second byte of the MVC instruction. Since this byte is all binary 0’s, the “Or” works like a copy. At run-time, the length byte contains x’03’ which causes 4 bytes to be moved into X. Remember that for **MVC**’s, the number of bytes in the object code is always 1 less than the number of bytes that the machine moves.

Examples

Some Unrelated EXs

```
R4 =  X'00000005'
R5 =  X'FFFFFF04'
R6 =  X'00000034'
```

	Instruction		**Object Code**
TARGET1	MVC	FIELDA(0),FIELDB	D2 00 C0 10 C0 20
TARGET2	AP	FIELDC(0),FIELDD(0)	FA 00 C0 22 C0 34
TARGET3	LR	R0,R0	18 00

```
            EX    R4,TARGET1  MOVES 6 BYTES FROM FIELDB TO FIELDA
            EX    R5,TARGET1  MOVES 5 BYTES (ONLY RIGHTMOST BYTE OF
R5
                              DETERMINES THE LENGTH)
            EX    R6,TARGET1  MOVE 53 BYTES (LENGTH IS IN HEX)
            EX    R6,TARGET2  THE SECOND BYTE OF AN AP CONTAINS TWO
                              LENGTHS.  FIELDD (5 BYTES) IS ADDED TO
                              FIELDC (4 BYTES).
            EX    R6,TARGET3  THE SECOND BYTE OF AN LR SPECIFIES 2
                              REGISTERS.  REGISTER 4 IS LOADED INTO
                              REGISTER 3.
```

Tips

1. The standard use for an execute instruction is to support variable length moves. The previous examples illustrate that instructions other than **MVC** can be executed but this should be carefully considered. For instance, in executing an **LR** instruction, the registers can be selected dynamically. This is probably not a wise choice and may result in a program that is very difficult to debug.

2. When coding the target instruction, be sure and specify an explicit length of zero so the second byte of the machine code is x'00'.

3. Place the target instruction in a place where it will never be executed except as the target of an **EX** instruction. Most programmers put the target instructions among their **DS**'s and **DC**'s.

4. While the target instruction can be any instruction except another **EX** instruction, you should limit the target instructions to **MVC**'s.