STY R_1,D_2(X_2,B_2)

Store Fullword

RXY-a

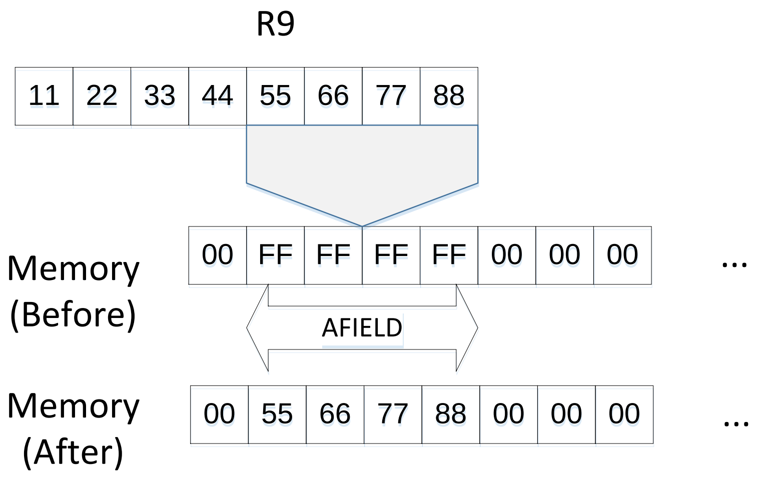
E3	R_1X_2	B_2DL_2	DL_2DL_2	DH_2DH_2	50

STY is used to copy the fullword stored in the register specified by operand 1 into the fullword memory location specified by operand 2. **STY** is similar to ST but provides a 20-bit signed displacement in the base/displacement address for operand 2, while ST provides a 12-bit displacement. As a result, operand two, the fullword in memory, must lie within 524,287 bytes beyond the location designated by a base register or base-and-index-register pair, or 524,288 bytes before that location. This is a much larger range than the 2^{12} = 4,096 byte range provided by ST.

STY has a two-byte opcode, E350.

Consider the following example,

```
STY     R9,AFIELD
```

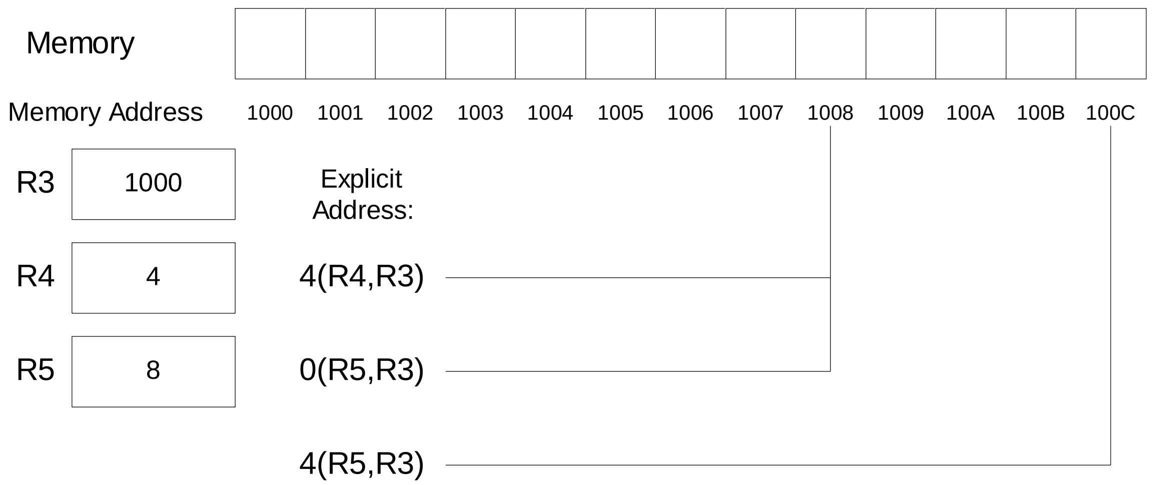


In this case, the contents of register 9 are copied to the fullword in memory denoted by AFIELD. This operation destroys the previous contents of AFIELD but leaves R9 unchanged.

STY provides an index register which may be coded as part of operand 2. Notice that in the previous example, no index register was specified. When the index register is omitted, the assembler chooses R0, which does not contribute to the address. The following example illustrates this idea.

```
STY      R9,AFIELD(R5)
```

The assembler converts AFIELD to a base register and displacement, while R5 is the index register. For instance, the expression AFIELD(R5) might (we cannot determine the base register) be equivalent to the explicit address 0(R5,R3) - displacement = 0, index register = R5, base register = R3. The effective address is computed by adding the base register contents to the index register contents plus the displacement. If the index register contains an “8”, then AFIELD(R5) refers to the fullword that begins at an 8 byte displacement from the beginning byte of AFIELD. The following examples illustrate several explicit addresses that include an index register.



In the first explicit address, 4(R4, R3), the effective address is computed by adding the contents of base register 5, the contents of index register 3, and the displacement (1000 + 4 + 4 = 1008). The second address 0(R5, R3) is computed as 1000 + 8 + 0 = 1008, and the third address, 4(R5, R3) is computed to be 1000 + 8 + 4 = 100C (hexadecimal).

If an index register is not explicitly coded, as in the instruction “STY R9,AFIELD”, the assembler chooses R0 as the index register, which does not contribute to the effective address.

Examples

Some Unrelated STY’s

```
               R7 =  X’0000000000001000’
               R8 =  X’0000000000000004’
               R9 =  X’0000000000000008’

  AFIELD   DC  F’20’      AFIELD = X’00000016’
  BFIELD   DC  F’-1’      BFIELD = X’FFFFFFFF’
  CFIELD   DC  F’0’       CFIELD = X’00000000’

           STY   R7,AFIELD     AFIELD = X’00001000’
           STY   R8,AFIELD     AFIELD = X’00000004’
           STY   R8,BFIELD     BFIELD = X’00000004’
           STY   R7,AFIELD(R8)  CHANGES BFIELD TO X’00001000’
           STY   R7,AFIELD(R9)  CHANGES CFIELD TO X’00001000’
```

Tips

1. Operand 2 should reference an aligned fullword in memory. It is possible to store the contents of a register into 4 bytes of memory that are not aligned on a fullword, but the assembler will warn you that operand 2 is not properly aligned.

2. You might consider using **STY** instead of ST in cases where you have maxed out a base register. Rather than adding another base register to fix an addressability error, consider using **STY** to help solve your problem.

3. Many RX instructions have companion instructions with the RXY-a instruction format . RXY-a instructions all provide 20-bit displacements (range 0 - 1,048,575) instead of 12-bit RX displacements (range 0 – 4095). For example, LY is the companion Load Fullword operation to L.