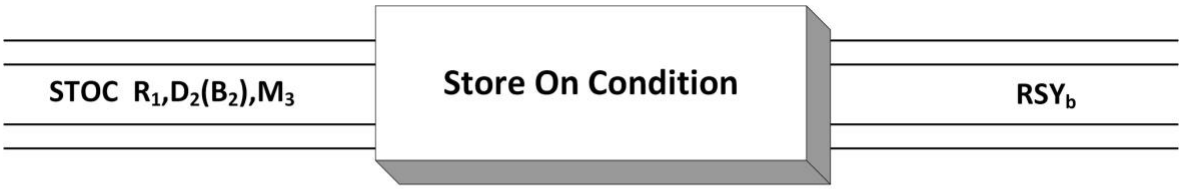


Store on Condition (**STOC)** stores (or not) the rightmost 32 bits of a register designated by R1 into a fullword in memory designated by operand 2. The fullword is stored in memory depending on a combination of the mask bits in operand 3, and the current condition code.

The mask field consists of four bits representing conditions equal/zero, low/minus, high/plus, and overflow in sequence from left to right. A 1 bit in the mask indicates the load should occur if the corresponding condition exists. For example, mask B'1000' represents condition equal/zero, while mask B'1010' represents conditions equal/zero or high/plus, and mask B'0111' represents conditions low/minus, high/plus, and overflow. If the current condition code matches any bit in the mask, the rightmost 32 bits of R1 are stored, otherwise no store occurs.

For example, assume the current condition code is high. Consider the following code,

```
          STOC    5,X,B'0111'
```

Since the high bit is indicated in the above mask (along with low/minus and overflow), the right 32 bits of register 5 are stored in fullword X. The above code is equivalent to the following lines,

```
          BE      THERE
          ST      5,X
THERE     EQU     *
```

Notice that control falls through the BE on conditions low/minus, high/plus, and overflow. As a result, the store operation occurs on these conditions.

The example above illustrates when to choose STOC over ST. If the condition code is set and the store is conditional based on the current value of the condition code, a judicious use of STOC can mean one less branch instruction in your program.

HLASM provides relief from having to code a binary mask by providing suffixes that can be appended to STOC to indicate the conditions under which the store should occur. An earlier version of HLASM provided the following suffixes.

Suffix	Condition	Effective M_3 Value
E	Equal	B'1000'
L	Low	B'0100'
H	High	B'0010'
NE	Not equal	B'0111'
NL	Not low	B'1011'
NH	Not high	B'1101'

Using these suffixes you can code this,

```
          STOCH  3,X
```

instead of this,

```
          STOC   3,X,B'0010'
```

The most current version of HLASM provides the following additional suffixes.

Suffix	Condition	Effective M_3 Value
Z	Zero	B'1000'
M	Minus	B'0100'
P	Plus	B'0010'
NZ	Not zero	B'0111'
NM	Not minus	B'1011'
NP	Not plus	B'1101'
NO	Not overflow	B'1110'

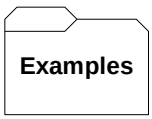
Below is an assembly listing of STOC instructions using all possible suffixes. The opcode, EBF3, occupies the first and last bytes in the object code. The operand 1 register is 3. The base displacement address C22600 (which when rearranged in proper sequence is C00226) is the base/displacement address of X. The only difference in these instructions occurs in the fourth digit of the object code and corresponds to the suffix appended to STOC.

```
  Loc  Object Code    Addr1 Addr2  Stmt   Source Statement

00002C EB38 C226 00F3       0022C    63          STOCE  R3,X
000032 EB34 C226 00F3       0022C    64          STOCL  R3,X
000038 EB32 C226 00F3       0022C    65          STOCH  R3,X
00003E EB37 C226 00F3       0022C    66          STOCNE R3,X
000044 EB3B C226 00F3       0022C    67          STOCNL R3,X
00004A EB3D C226 00F3       0022C    68          STOCNH R3,X
000050 EB38 C226 00F3       0022C    69          STOCZ  R3,X
000056 EB34 C226 00F3       0022C    70          STOCM  R3,X
00005C EB38 C226 00F3       0022C    71          STOCE  R3,X
000062 EB34 C226 00F3       0022C    72          STOCL  R3,X
```

```
000068 EB32 C226 00F3       0022C    73          STOCH  R3,X
00006E EB3B C226 00F3       0022C    74          STOCNL R3,X
000074 EB3D C226 00F3       0022C    75          STOCNH R3,X
00007A EB38 C226 00F3       0022C    76          STOCZ  R3,X
000080 EB34 C226 00F3       0022C    77          STOCM  R3,X
000086 EB32 C226 00F3       0022C    78          STOCP  R3,X
00008C EB37 C226 00F3       0022C    79          STOCNZ R3,X
```

There are two binary masks for which there is no corresponding suffix. B'0000' would indicate that the store should never occur, and if no exceptional condition exists, makes the STOC function as a NOP. B'1111' indicates the store should always occur, and if no exceptional condition exists, makes the STOC function as ST. In both cases it is better to avoid using STOC.



Assume the following declaration for all the examples below,

```
Y         DC   F'100'

          STOC    R6,Y,B'0010' STORE R6 INTO Y IF CC = HIGH/PLUS
          STOCH   R6,Y         EQUIVALENT TO THE INSTRUCTION ABOVE

          STOC    R9,Y,B'1011' STORE R9 INTO Y IF CC NOT = LOW/MINUS
          STOCNL  R9,Y         EQUIVALENT TO THE INSTRUCTION ABOVE

          STOC    R7,Y,B'1000' STORE R7 INTO Y IF CC = EQUAL/ZERO
          STOCZ   R7,Y         STORE R7 INTO Y IF CC = EQUAL/ZERO
          STOCE   R7,Y         STORE R7 INTO Y IF CC = EQUAL/ZERO

          STOC    R7,Y,B'0111' STORE R7 INTO Y IF CC NOT = EQUAL/ZERO
          STOCNZ  R7,Y         EQUIVALENT TO THE INSTRUCTION ABOVE
```

Tips

1. Use STOC instead of ST in situations where the store occurs conditionally. This avoids coding an additional branch instruction.

2. Avoid the use of masks B'0000' and B'1111' with STOC. Choose NOP and ST instead.

3. LOC is the sister instruction to LOC. It works in an analogous way to STOC but loads a fullword in memory into a register instead of storing.

4. STOC and LOC are only available on certain models. Check the Principles of Operation for your machine.

5. Consult the Principles of Operation for situations in which STOC can provide significant performance improvement and for which models of the machine support this instruction.