SPM R_1

Set Program Mask

RR

04	R_1//

The Set Program Mask instruction, **SPM**, is used to change the 2-bit condition code and 4-bit program mask in the current PSW. Before executing **SPM**, you should set bits 34 and 35 of general register R_1 to the desired value of the condition code. You should also initialize bits 36-39 of general register R_1 – these values will replace the program mask. Bits 0-33 and 40-63 of general register R_1 are ignored. Here is an example:

```
         L    R8,WORD1     SET REG 8 WITH CC = 3 BITS
         SPM  R8           MODIFY THE CURRENT PSW
         …
WORD1    DC   X’30000000’  CONDITION CODE = B’11’ (HIGH)
                           PROGRAM MASK = B’0000’
```

The program mask bits occur in bits 20-23 of the current PSW. These are the bits modified by **SPM**. Each bit represents a type of interruption:

PSW Bit 20 - Fixed-point overflow
PSW Bit 21 - Decimal overflow
PSW Bit 22 - HFP exponent underflow
PSW Bit 23 - HFP significance

If a program mask bit is set to 1, an exception results in the occurrence of the corresponding interruption, and the program will abend. If a bit is set to 0, the corresponding interruption is suppressed. For example, suppose bit 21 (Decimal overflow) in the PSW is 0, and we execute the following instructions,

```
        AP X,=P’1’
        BO THERE
        …
X       DC  PL2’999’
```

Adding one causes a decimal overflow in operand 1 (1000 doesn’t fit in a two byte packed field), but the program mask bit being 0 causes the interrupt to be suppressed. Our program doesn’t abend, and as a result, we can detect the overflow with a Branch On Overflow (BO) mnemonic on the next line. On the other hand, if the program mask bit in the PSW had been 1, the interrupt would have occurred and our program would have abended before reaching BO.

While the initial settings of the program mask bits are typically B’0000’, it is possible that your installation could change these to different default values. You can use IPM to retrieve the initial values if you are unsure, or you can simply try the code above and see if the branch is taken.

Examples

Some Unrelated SPMs

Assume the following fields:

```
          DS 0F
WORD1     DC X’30000000’    CC = B’11’, PROGRAM MASK = B’0000’
WORD2     DC X’1F000000’    CC = B’01’  PROGRAM MASK = B’1111’
WORD3     DC X’0C000000’    CC = B’00’  PROGRAM MASK = B’1100’


        L      R4,WORD1
        SPM    R4           CC = B’11’, PROGRAM MASK = B’0000’

        L      R5,WORD2
        SPM    R5           CC = B’01’  PROGRAM MASK = B’1111’

        L      R9,WORD3
        SPM    R9           CC = B’00’  PROGRAM MASK = B’1100’
```

Tips

1) **IPM** (Insert Program Mask) is the companion instruction to **SPM** that retrieves information from the current PSW and stores it in a register.

2) Programmers doing business arithmetic often have to worry about decimal overflows. **SPM** provides us with two strategies for dealing with these:

   a) Turn off interrupts and check for overflows with branching logic. This strategy could add a significant number of tests since decimal overflows can occur on each AP, SP, and ZAP. (There are a number of fixed point instructions which can produce an overflow.)
   b) Turn on interrupts as a way of catching errors and omit the testing logic for overflows.

   This also leads to the possibility of using the first strategy during program development and using the second strategy during production.