SRST R_1,R_2

Search String

RRE

OP Code	OP Code	//	R_1R_2

SRST is an RRE instruction which can be used to sequentially search for a single character in a contiguous area of memory. The instruction has the ability to search to the top of memory and then wrap around and continue the search at address 0. Another complicating factor associated with **SRST** is that if the area of memory being searched is greater than 256 bytes, the CPU can arbitrarily halt the search for efficiency reasons. In this case, the programmer can continue the search easily by looping back and restarting it. This feature requires that when searching large areas of storage (> 256 bytes), programming logic must be supplied to deal with possible interruptions by the CPU. When searching smaller areas of storage (< = 256 bytes), no interruptions can occur. In spite of some complicating conditions, this instruction is relatively easy to use.

This instruction was no doubt added to the instruction set to give some relief to compiler writers who needed to find null-terminated strings in languages like C. Using this instruction, it is possible to search for the first occurrence of X'00' in a contiguous collection of memory, thus identifying the end of a C-string.

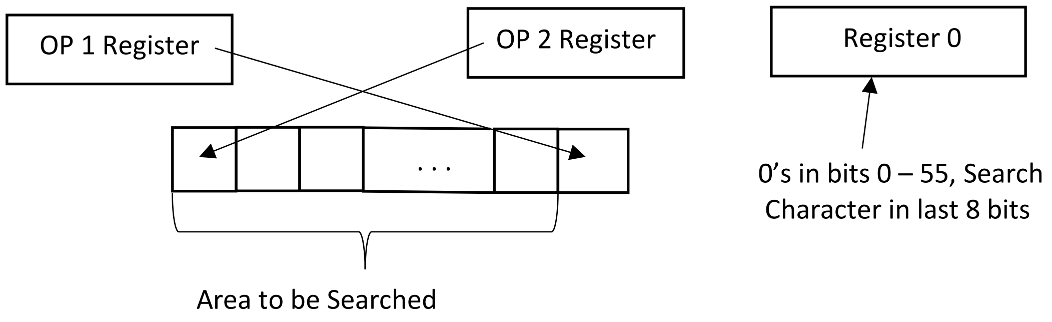
This instruction has two operands, both registers, and it behaves differently depending on the addressing mode. For this discussion, we assume 31-bit addressing mode with addresses being stored in bit positions 33-63 of each 64-bit register.

Prior to executing SRST, we establish the following conditions:

- The operand 2 register should contain the address of the first byte of memory that is about to be searched.

- The operand 1 register should contain the address of the first byte after the area memory that is about to be searched.

- The character that you are searching for should be stored in bit positions 56-63 of general register 0. All other bits in that register should be 0s.



The search proceeds from lower-numbered bytes to higher-numbered bytes, and continues until one of the following conditions occurs:

1. **The specified character has been found.** This is indicated with condition code 1. Test this with BL. The operand 1 register is updated with the address of the desired byte. Operand 2 is unchanged.

2. **The specified character is not found.** This is indicated with condition code 2. Test this with BH. Operands 1 and 2 are unchanged.

3. **The search is interrupted by the CPU.** This is indicated with condition code 3. Test this with BO. Operand 2 (which contains the beginning byte of the contiguous storage area we are searching) is updated to point at the first byte that has not yet been searched. Operand 1 is unchanged. The programmer can choose to continue the search simply by branching back on condition 3 to the SRST instruction.

Here is an example which uses SRST to find the first occurrence of a character "A" within a string.

```
         LA   R5,SRCAREA   OP2 POINTS AT START OF AREA
         LA   R4,AREAEND   OP1 POINTS ONE BYTE BEYOND AREA END
         LA   R0,C'A'      R0 CONTAINS 0'S WITH SRCH CHAR IN LAST BYTE
LOOP     DS  0H
         SRST R4,R5        SEARCH FOR AN 'A'
         BC   3,LOOP       IF SEARCH INTERRUPTED, RESTART IT
         BC   2,NOTFND     CHAR WAS NOT FOUND
FOUND    DS  0H            CHAR WAS FOUND
         ...               R5 CONTAINS THE ADDRESS OF FIRST 'A'
                           R4 IS UNCHANGED
NOTFND   DS  0H
         ...
SRCAREA  DS  CL1000
AREAEND  EQU *
```

One unusual feature of this instruction is that the CPU may interrupt the search. This is signaled with condition code 3. When that occurs, the operand 2 register is updated with the address of the first byte that hasn't been examined. This means that the search can easily be continued by branching back to the instruction. This restarts the search using the new starting address. As a result, it's not uncommon to test the condition code after each search, and branch back to restart it again if the condition code is 3. That idea is illustrated in the code above. This is unnecessary if all the searches are conducted on areas that are <= 256 bytes.

It's possible to conduct a search without knowing the ending address of the area you are searching:

1. Load register 0 with character you are trying to find.
2. Load the operand 1 register with 0. This effectively makes the ending address occur at byte 0 after wrapping around the upper byte of memory.
3. Load the operand 2 register with the address of the first byte of the area that is to be searched.

Register 0 can be used (under certain conditions) as operand 1 or operand 2. Consult the Principles of operation for the details.

Examples

Two unrelated SRST's:

```
         LA   R0,X'FF'              NEED TO SEARCH FOR HIGH VALUES BYTE
         L    R7,=A(SRCHAREA)       R7 POINTS AT AREA TO BE SEARCHED
         L    R3,=A(SRCHAREA+1000)  R3 POINTS 1000 BYTES AFTER THAT
LOOP     SRST R3,R7                 SEARCH 1000 BYTES OF AREA FOR X'FF'
         BC   3,LOOP                CONTINUE IF SEARCH INTERRUPTED
         BH   NOTFND                CC = 2 - BYTE NOT FOUND
          ...                       X'FF' WAS FOUND, R7 HAS THE ADDRESS
NOTFND   EQU  *                     X'FF' WAS NOT FOUND
```

```
         SR   R0,R0                 SEARCH FOR X'00'
         LA   R5,SRCHAREA           R5 POINTS AT THE SEARCH AREA
LOOP     SRST R0,R5                 SEARCH FOR X'00',STOP AT BYTE 0
         BO   LOOP                  CONTINUE IF SEARCH INTERRUPTED
         BH   NOTFND                CC = 2 - BYTE NOT FOUND
          ...                       X'00' WAS FOUND, R7 HAS THE ADDRESS
NOTFND   EQU  *                     X'00' WAS NOT FOUND
```

Tips

1. When searching areas that are larger than 256 bytes, always test for condition 3 after the search.
2. `BO LOOP` is equivalent to `BC   3,LOOP`
3. `BL LOOP` is equivalent to `BC   1,LOOP`
4. `BH LOOP` is equivalent to `BC   2,LOOP`