S/390 Assembler Instruction Set

by Paul Stuyvesant

If you are a TPF Programmer who works mainly with Assembler you could be forgiven for thinking that not a lot has changed in the last 10 (or 20) years. So, we have TPF 4.1 using DAT, large amounts of memory, VFA and so on. But what has changed in the Assembler world? Not an enormous amount, however the change to S/390 introduced some new instructions that the assembler programmer can use. If you have a Yellow, Salmon or Blue Reference Summary it would be worth getting hold of the latest copy, which is of course - White. Available from all good bookshops or online at the IBM website.

So why use the new instructions? The Halfword Immediate instructions are useful in that you do not need storage for the halwords. The string instructions are very useful if you are storing data in C programs using the null terminator (X’00’) to show the end of the string. But that is not their only use. They can also cut down on the Execute instruction, as they deal with a specific character in your data which you can set up yourself. But first a couple of provisos.

1. All processors that the code is going to run on must support the new instruction set, so if you are using older processors, maybe for a test system or fallback machine, these must also be S/390 compliant.

2. If you are developing code that may be sold on to other customers, care should be taken when using these instructions. If the customer does not have S/390 processors the programs will not run on their systems.

Halfword Immediate

This group of instructions all work with halfwords. Unlike the usual set of instructions such as LH and CH which work with fields in store, the new instructions work with an Immediate halfword that is held in the machine code.

These Instructions are Immediate and Relative (RI) format. Installation is optional so it needs to be checked that they are installed on the system you are working on.

Op Code	R_1	Op Code	I_2

0 8 12 16 31

The Op-code for each of these instructions is 1½ bytes in length, the final 4 bits of the Op-code coming from bits 12-16 of the machine code.

Example	Machine Code	Use
AHI R2,H'12' AHI R3,X'00BC'	A7 2A 000C A7 3A 00BC	Op-Code A7A. Add the Immediate halfword to the contents of the 1st Operand register. CC0 Result is Zero, No Overflow CC1 Result is -ve, No Overflow CC2 Result is +ve, No Overflow CC3 Overflow
LHI R2,H'36'	A7 28 0024	Op-Code A78. Load the Immediate halfword (sign extended) to the specified register. The Condition Code is not set for this instruction

Example	Machine Code	Use
CHI R6,H'12'	A7 6E 000C	Op-Code A7E. Compare the Immediate halfword to the contents of the register. CC0 Equal CC1 1st Operand low CC2 1st Operand high CC3 Not used
MHI R2,H'36'	A7 2C 0024	Op-Code A7C. Multiply the Immediate halfword by the specified register. Only the rightmost 32 bits of the result are placed in the register and any overflow is ignored. The Condition Code is not set for this instruction.

String Instructions
Also available are a group of string instructions. These are similar to the MVC/CLC instructions except that there is no length parameter. The programmer supplies an end character in R0 and this is used to identify the end of the string. For these instructions the end character of the data to be moved must be known, and if used they will reduce the need for Execute instructions when working with variable length fields.

These instructions are very useful if data is being stored using the C language, with X’00’ (the null terminator) as the end character of a text field.

In all of these instructions bits 16-23 of the machine instruction are ignored. The number of bytes processed is CPU-determined and is at least 256 bytes. All MVST, CLST and SRST instructions are RRE format.

Note: Bits 24-31 of R0 should contain the specified character for these instructions. Bits 0-23 of R0 are reserved for future use and should always contain 0.

Example	Use

```
      SR   R0,R0
      LA   R0,#EOM
      LA   R6,SOURCE
      LA   R7,DEST
LAB1  EQU  *
      MVST R7,R6
      BC   1,LAB1
* data moved until #EOM found
```
	The Move String (MVST) is used to move a string of data.
	
	MVST. Op-Code B255. Move String. Moves the data from SOURCE to DEST until the X'4E' is found in the SOURCE, or a CPU determined number of bytes is moved if the X'4E' is not found.
	
	When the resulting Condition Code is 3 the program can simply branch back to the MVST instruction, as the 2 operands will now contain updated addresses of the next bytes to be processed.
	
	CC0 Not set
	CC1 Entire string moved. 1st op register contains address of ending character in 1st op. Op 2 is unchanged.
	CC2 Not set.
	CC3 Number of bytes moved determined by CPU. Both operand registers updated with address of next bytes to move.

Example	Use
SR R0,R0 LA R6,FLD1 LA R7,FLD2 LAB1 EQU * CLST R7,R6 BC 1,LAB1 * Complete string checked * Other processing here	The Compare Logical String (CLST) is used to compare two strings of data. CLST. Op-Code B25D. Compare String. Compares the 2 operands until unequal bytes are found. Compares the data from FLD1 with FLD2 until the X'00' is found in the SOURCE, or a CPU determined number of bytes is compared if the X'00' is not found. CC0 Entire operands equal. Both operands unchanged. CC1 first operand low. Both operands updated with addresses of last bytes processed. CC2 first operand high. Both operands updated of addresses of last bytes processed. CC3 Number of bytes compared determined by CPU. Both operand registers updated with address of next bytes to compare.
SR R0,R0 LA R0,X'D7' LA R6,START LA R7,1000(R6) LAB1 EQU * SRST R7,R6 BC 1,LAB1 BC 4,FOUND * Not found code here * FOUND EQU * * R7 points to found character	The Search String (SRST) is used to search a string of data for a specific character supplied by the programmer. This can be used to replace loops that check each byte individually. SRST. Op-Code B25E. The 2nd operand is searched for the character specified in R0. The 1st operand register contains the address where the search is to finish. The string is searched until the character is found or a CPU determined number of bytes are checked. This search will be at least 256 bytes. CC0 Not set. CC1 Specified character found. Op 1 contains address of character found. Op 2 is unchanged. CC2 Specified character not found. Both operands unchanged. CC3 Number of bytes compared determined by CPU. Operand 1 is unchanged. Operand 2 is updated with address of next byte.

Bibliography

IBM have put most of the manuals online. At the moment they are available in PDF format (requires Adobe Acrobat) at: www.s390.ibm.com/os390/bkserv/r10pdf/os390_sys.html and the books you need for the new instructions are:

Name	**IBM File Number**
ESA/390 Reference Summary	SA22-7209-02
ESA/390 Principles of Operation	SA22-7201-06

Paul Stuyvesant runs PCS Training, a company specialising in TPF, Assembler and ISOC training. If you have any questions on the instructions covered in this article he can be contacted at paul@pcs-training.co.uk. A list of available courses can be viewed on the PCS Training website, www.pcs-training.co.uk.