USING EXTRA DATA SEGMENTS: SAFE AND EFFICIENT

Rick Ehrhart
Hughes Aircraft Company, El Segundo
M/S 335/510
P.O. Box 92426
Los Angeles, CA 90009
(213) 648-0755

Abstract

Handling extra data segments on the HP 3000 has given rise to numerous lectures, papers and classes. The methodology has always been to use the data segment "DS" intrinsics with the extra data segment "DS" capability. The purpose of this paper is to demonstrate a new methodology, one using the privileged "PM" capability.

Introduction

There are two types of data segments on the HP 3000: the stack and the extra data segment. There is one and only one stack for each process to utilize the data in a predefined structure. The extra data segment, on the other hand, can be utilized by one or more processes in a user defined structure. It may be used as a table or an area for process-to-process communication.

For most applications, the "DS" instrinsics are quite adequate, but when the programmer is dealing with system level processes, the overhead of the "DS" intrinsics is unnecessary. This paper will show how to use three very powerful and privileged assembly level instructions that work on data segments: MTDS, MFDS, MDS.

Techniques

To acquire an extra data segment, the programmer still must use the GETDSEG intrinsic. In non-privileged mode, GETDSEG returns a logical index; but in privileged mode, GETDSEG returns the Data Segment Table Index or "DSTX". This value is the unique index for the extra data segment that the process acquires. All data segments, whether a stack or an extra data segment, have a unique DSTX. The DSTX must be saved for all further operations.

To move data into the extra data segment from the stack, the programmer uses the "DS" intrinsic DMOVOUT. DMOVOUT checks the boundaries of both the stack and the extra data segment and then moves the data out from the stack to the extra data segment. However, in privileged mode, the programmer doesn't have to use DMOVOUT, he/she may use the assembly instruction MTDS, Move To Data Segment. This one instruction moves words from a DB-relative location to the extra data segment starting at a specified offset for a count. Usage is straightforward and simple. Example One has a procedure that shows how to use the MTDS assembly instruction. Basicly, the following items are pushed onto the top of the stack: target DSTX, the one returned from GETDSEG, the target offset, the source DB-relative address, where the data is in the stack, and a positive count of the number of words to be moved. The one drawback is the fact that the programmer has to make sure that the DSTX, the offset, the DB-relative address, and the count, are all valid.

To move data into the stack, the programmer used to call DMOVIN; but now the programmer may use the second privileged assembly instruction MFDS, Move From Data Segment. This instruction expects the following values on the top of the stack: the target DB-relative address, the source DSTX, the source offset in the DSTX, and a positive count. To see how to use the MTDS instruction, please see Example Two.

The third privileged assembly instruction is MDS, Move using Data Segments. It moves data from one data segment to another. This instruction requires the following values on the stack: the target DSTX, the target offset, the source DSTX, the source offset, and a positive or negative count. Please see Example Three for usage.

To release the extra data segment, the programmer uses the FREEDSEG intrinsic. The DSTX returned from GETDSEG is used as the index. The process has to be in privileged mode to call FREEDSEG.

Conclusion

It is highly recommended that the programmer read the HP 3000 SERIES 2 MACHINE INSTRUCTION SET REFERENCE MANUAL, Part. No. 30000-90022 before attempting to use these privileged assembly instructions, since with these instructions, the programmer has a chance to destroy the system's integrity in one swift blow. For example, if the DSTX is invalid, the system will come to a halt with system failure 16; or if the DSTX is wrong, the instruction could overlay another user's stack or even a system table.

The above techniques do cut down on overhead when working with extra data segments. This is very useful for critical systems like a communi-cations system, on-line monitor, or where processes have to communicate very quickly. These techniques have been used at Hughes Aircraft Company quite successfully, and the benefits are well worth the dangers.

```
<<**>>  EXAMPLE  ONE  <<**>>
<<------------------------------------------------------------------>>
<<>> PROCEDURE MOVE'TO'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE;
 TOS := COUNT;
 ASSEMBLE(MTDS 4);
 END;  << MOVE'TO'XDS >>


<<**>>  EXAMPLE  TWO  <<**>>
<<------------------------------------------------------------------>>
<<>> PROCEDURE MOVE'FROM'XDS(TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT);
     VALUE TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MFDS 4);
 END;  << MOVE'FROM'XDS >>


<<**>>  EXAMPLE  THREE  <<**>>
<<------------------------------------------------------------------>>
<<>> PROCEDURE MOVE'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,
                       SOURCE'OFFSET,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MDS 5);
 END;  << MOVE'XDS >>
```

```
<<**>>  EXAMPLE  ONE  <<**>>
<<---------------------------------------------------------------->>
<<>> PROCEDURE MOVE'TO'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE;
 TOS := COUNT;
 ASSEMBLE(MTDS 4);
 END;  << MOVE'TO'XDS >>


<<**>>  EXAMPLE  TWO  <<**>>
<<---------------------------------------------------------------->>
<<>> PROCEDURE MOVE'FROM'XDS(TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT);
     VALUE TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MFDS 4);
 END;  << MOVE'FROM'XDS >>


<<**>>  EXAMPLE  THREE  <<**>>
<<---------------------------------------------------------------->>
<<>> PROCEDURE MOVE'XDS(TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,
                        SOURCE'OFFSET,COUNT);
     VALUE TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     INTEGER TARGET'DSTX,TARGET'OFFSET,SOURCE'DSTX,SOURCE'OFFSET,COUNT;
     OPTION PRIVILEGED;
BEGIN
 TOS := TARGET'DSTX;
 TOS := TARGET'OFFSET;
 TOS := SOURCE'DSTX;
 TOS := SOURCE'OFFSET;
 TOS := COUNT;
 ASSEMBLE(MDS 5);
 END;  << MOVE'XDS >>
```