To: Distribution

From: Steve Webber

Subject: 6180 hardware modifications

Date: 1/2/74

Introduction

During the past year several hardware modifications have been proposed to the 6180 processor to facilitate implementation of user-ring subsystems (linkers, on-line T and D, etc.) as well as to simplify the standard workings of the system. Some of these changes appear to be useful and not too difficult to implement. A list of these is given below so that the reader may comment on their usefulness, feasibility and whether they preserve the security of the system as it stands.

(1) Redefine the "read bracket"

The current definition of the read bracket is [r1, r2], i.e. a procedure in ring R can read a segment with such brackets if r1 <= R <= r2. The proposed change is to make the read bracket [r1, r3]. This, of course, will differ from the current definition only if r2 ^= r3, i.e. the segment is a gate. In this case the gate segment would be readable in the rings from which it can be called but not executed.

There seems to be no problem with doing this since the gates can always be organized to have as little or as much information in them as desired. In the normal operating mode, a gate will consist of a transfer vector followed by appropriate definitions for the gate entries and targets. This definition information (as well as the linkage and symbol information) would be available to the caller under the new scheme. There seems to be no problem with this. Indeed, if we were to implement a linker in rings other than 0 the linker would like to be able to read just this information.

The hardware changes for this should be very slight assuming there is a little free space on boards affected.

(2) Packed pointer load fault.

It would be very convenient (in particular for a user-ring linker) to have a hardware fault generated if a packed pointer load encountered a particular bit pattern. It is currently possible to cause a fault when loading a potential ITS pointer by

replacing the ITS tag with a fault tag. (This is because the only convenient way to load an ITS pointer is with an indirect EPPn instruction.) The same feature would be desirable for packed pointers.

The hardware currently must check the segment number when loading a packed pointer so that it can extend the segment number 7777(8) to 77777(8). What is needed is for the hardware to also check the bit field for, say, 77(8) (an illegal bit offset) and fault in such a way that the packed pointer could be replaced by a legitimate pointer by the fault handler and a return made to the original LPPPn instruction which failed.

I understand the hardware that faults on SPRPn if a segment number is greater than 7777(8) is still being worked on and that this may make it easier to implement the proposed change.

(3) Redefine meaning of bit 28 in normal mode.

The third change to the hardware that is proposed is to make the meaning of bit 28 of an instruction (the inhibit bit) the same in all operating modes of the processor. Currently, in the "normal" mode (not privileged, not BAR) the inhibit bit has no effect. This turns out to be an inconvenience as several new uses of the inhibit bit have come to light. The original reasons for making the inhibit bit have no effect are not strong enough to override the current potential uses.

This change would simplify the hardware and is apparently easy to implement.

There will have to be a slight change made (any time before the hardware is changed) to the Multics signalling mechanism to prevent a user from running entirely in inhibited code. (The lockup fault that gets signalled if a user runs inhibited too long must cause some uninhibited code to be executed in the user ring.)

(4) Decrease number of allowed segment numbers.

It has been pointed out that with the use of packed pointers segment numbers greater than 7777(8) are inconvenient to use and regulate. Further, for normal Multics running it is rare that under <u>current</u> uses of the system that a process ever has more than several hundred segments known at the same time. There is currently a limit of 1024 segments in the KST and a limit of 2048 segments in the descriptor segment.

It is therefore proposed that the segment number field in pointer registers, ITS indirect words and in all relevent internal processor registers be reduced to 12 bits. This simplifies hardware in loading and storing of packed pointers which must look out for null pointers as well as look out for storing a pointer in packed form whose segment number is greater than 7777(8). The change also removes the possibility of trouble when and if the system ever tries to use more than 7777(8) segment numbers at once by making it impossible to do so.

This change also simplifies the ring 0 software which would

have to disallow the segment number 7777(8).

(5) Save and restore indicators across a call.

The 645 processor had the feature that if a standard call and return mechanism is used, the processor indicators are preserved across the call. This feature was not thought to be useful and therefore the 6180 specifications did not call for the feature to be supported. It turns out, however, that it is necessary to save the indicators in order to preserve a rigorous PL/I environment and we have therefore been forced to implement the preservation of the indicators with software.

It is therefore proposed that the hardware be modified in such a way that the STCD and RTCD instructions save and restore the indicators in the same manner that was done on the 645 hardware.

The hardware changes necessary to do this are unfortunately fairly extensive in the case of the STCD instruction. The RTCD falls out very easily as they had to go to special effort to stop it from doing what we now want. It turns out that the standard call operator used by PL/I and FORTRAN does not make use of the STCD instruction anyway so just changing the RTCD would be quite beneficial.

(6) Clarify bit offset workings with respect to ITP

The bit offset that should be used when an ITP indirect reference is made should be formed by adding the bit offset of the selected pointer register to the bit offset specified in the ITP indirect word pair. The add may generate a carry into the word field of the effective address. If this is too difficult to implement, the hardware should use the bit offset specified in the ITP word rather than the bit offset of the selected pointer register.