PDP-10 VIRTUAL MACHINES

S. W. Galley
Project MAC, M.I.T.

ABSTRACT. The architecture of the DEC PDP-10 prohibits using standard methods to construct a virtual PDP-10 on it. However, by simulating all executive-mode instructions and in rare cases user-mode instructions, a "hybrid" virtual machine may be constructed. Building on such a simulator and an operating system with hierarchical process relationships, the author has easily constructed a prototype virtual machine.

CP-67 [ref 1] represents the standard method for implementing virtual machines on a third-generation computer. The success of the implementation depends on mapping the virtual processor's "supervisor state" or "executive mode" into the real processor's "problem state" or "user mode", with the guarantee that all sensitive instructions are privileged [ref 2]. In other words any instruction which would cause problems in interpretation or control if executed directly (sensitive) causes a trap to the supervisor if executed in problem state (privileged). The supervisor then arranges for the sensitive instruction to be simulated.

The PDP-10 computer [ref 3] has a few instructions which are sensitive but not privileged. These sensitive instructions alter or query the processor's user-mode bit, which signifies whether the processor is in executive mode or user mode. In particular the instructions jump to subroutine (JSR), jump and save program counter (JSP), and push down and jump (PUSHJ) query the value of the bit; the instruction jump and restore (JRST) can change the state from executive mode to user mode. Thus a virtual-machine map cannot map virtual executive mode into real user mode; the supervisor must simulate all instructions executed in virtual executive mode.

In considering virtual user mode, we must distinguish two sub-modes of user mode: standard user mode and "user in-out mode". In the latter mode input/output and certain other instructions, which are sensitive, are not privileged. Another processor bit, which can be queried by the sensitive instructions listed above, signifies the presence of user in-out mode. All other sensitive instructions are privileged, so a virtual-machine map can map virtual standard user mode into real user mode (executed directly) and virtual user in-out mode into simulated user mode. (In practice user in-out mode is rarely used. The instruction to enter user in-out mode is privileged.) The virtual machine created by a virtual-machine map that specifies simulation for virtual executive mode has been called a "hybrid" virtual machine [ref 2] to distinguish it from a standard one, as in CP-67.

The PDP-10 computer system of the Dynamic Modeling/ Computer Graphics group of Project MAC has two features that make the construction of a hybrid virtual machine relatively easy. The first is a feature of the time-sharing operating system in use, ITS [ref 4]. ITS organizes (user-mode) processes into trees, which can grow downward from the top-level process. A process can create an inferior process and can read and write its control block ("system variables") and memory space. One bit in the process control block determines whether or not the process is in "user-trap mode". When a process in user-trap mode executes a privileged instruction -- usually a call to the executive-mode operating system for service -- ITS does not perform the service itself but rather informs the immediately-superior process and allows it to perform the service for its inferior. (This feature of ITS was implemented to provide for simulation of the environments of other operating systems.) In our case the user-trap feature allows us to implement the virtual-machine monitor as a user-mode process. When the virtual machine is in standard user mode, it executes directly in a process in user-trap mode immediately inferior to the virtual-machine monitor; when it enters executive mode or user in-out mode, the virtual-machine monitor takes over and simulates each instruction until the virtual machine returns to standard user mode. The ITS operating system takes care of scheduling and real resource allocation, and the tasks of the virtual-machine monitor are greatly simplified. Since the virtual-machine monitor operates in an environment provided by ITS, with an instruction set extended beyond that of the bare processor, we call this implementation an "extended-host" virtual-machine monitor instead of a "bare-host" virtual-machine monitor.

The second feature that made the construction of a hybrid virtual machine relatively easy was the existence of a PDP-10 simulation program. This simulator was designed and implemented by the author to simulate execution of a user-mode program in connection with a graphical debugging package [ref 5]. It was straightforward to extend the simulator so as to handle an executive-mode program. The simulator simulates a standard KA-10 CPU and the standard PDP-10 input/output devices (teleprinter and paper tape). The preexistence of the simulator reduced to about one person-month the work required to implement a hybrid virtual machine.

The DM/CG computer installation is suitable for PDP-10 virtual-machine development because it is a research environment rather than a production environment. Development and debugging of the operating system ITS and other executive-mode software continues, seemingly forever, while ITS users -- including students and users at other ARPA Network sites across six time zones -- would like time-sharing to be available as much as possible. "Stand-alone" debugging time concurrent with user time would indeed be a boon. Another favorable factor is the availability of CPU time. The highly interactive nature of DM/CG work produces CPU utilization somewhat less than unity, even in prime time. Thus the relatively large amounts of CPU time needed to simulate virtual executive mode completely are available for use.

REFERENCES

1. "A Virtual Machine Time-sharing System", R. A. Meyer and L. H. Seawright, IBM Systems Journal, vol. 9 no. 3 pp. 199-218 (1970)

2. "Architectural Principles for Virtual Computer Systems", Robert P. Goldberg, report no. 24-72, Center for Research in Computing Technology, Harvard University, Cambridge, Mass. (November 1972)

3. "PDP-10 System Reference Manual", DEC-10-HGAC-D, Digital Equipment Corporation Program Library, Maynard, Mass. (August 1969)

4. "ITS 1.5 Reference Manual", D. E. Eastlake et al, Memo no. 161A, Artifical Intelligence Laboratory, M.I.T., Cambridge, Mass. (July 1969)

5. "Debugging using ESP -- Execution Simulator and Presenter", S. W. Galley, Document no. SYS.09.01, DM/CG group, Project MAC, M.I.T., Cambridge, Mass. (November 1971)