# Hardware Additions to Microprocessor Architecture Aid Software Development

M. W. Sievers
Communications Systems Research Section

*Simple additions to a microprocessor's architecture provide a programmer with two powerful debugging aids. These aids are useful both for initial software development and for routine system integrity diagnostics. One of these aids may be expanded into a virtual machine system.*

## I. Introduction

The job of a hardware engineer is to design hardware, which, all too often, is done at the expense of the software engineer who must use the hardware. Whether through oversight or done intentionally to save space and hold down costs, hardware frequently lacks features not specifically required but that could simplify the task of the software engineer.

When designing microprocessor systems, whether they be simple dedicated controllers or complex general-purpose systems, the basic architecture can be constructed with increasingly fewer integrated circuit packages. A hardware engineer mesmerized by the simplicity of the architecture will put his design efforts into building interfaces for the devices the system must communicate with. The hapless programmer may be fortunate enough to get displays and single-step features to help him debug his software—but these are barely adequate when long and complex programs are being checked out.

In this article an Address Trap (breakpoint) mechanism and last-in-first-out (LIFO) Address Stack are suggested as two additions to the basic microprocessor architecture whose functions are solely to aid the programmer. These devices provide the programmer with the ability to specify address breakpoints and to trace program execution back through $N$ instructions, where $N$ is the depth of the stack. Both devices, plus interface logic and buffering, have been designed for an INTEL 8080-based system using approximately 25 integrated-circuit packages.

Section V is devoted to a proposal for implementing a microprocessor virtual machine system via data and address traps. The interested reader not familiar with virtual machine concepts should consult Refs. 1–7.

## II. Basic Architecture

Consider Fig. 1 in which a basic microprocessor architecture is illustrated. Three buses, the Address Bus, Data Bus, and Control Bus, interconnect the Central

Processing Unit (CPU) and the various devices on the buses. This configuration is similar to the PDP-11 UNIBUS (Ref. 8) in which any device may be treated as memory or input–output at the hardware designer's option.

The Address Trap and Address LIFO are connected to the bus structure as shown. The Address Trap generates a one-bit DISABLE flag which is used to disable memory. This flag bit could properly be considered part of the Control Bus but is separated for clarity.

## III. Address Trap

Frequently a programmer debugging new software desires to know if a given address in his program space is accessed. Further, it is often desirable to check partial results in the calculation of a complex function. In either case, the basic microprocessor instruction set does not lend itself to performing these tasks without considerable overhead.

An Address Trap which causes an interruption in the normal program stream is ideal for implementing the features described above. It is a very simple hardware device that jams an instruction on the Data Bus when the Address Bus contains an address equal to the one stored in a register in the trap. The advantage this device affords over inserting patches into the code being executed is that since the user program is left intact, no overhead is required for keeping track of where patches are made.

A block diagram of an Address Trap is shown in Fig. 2. The trap is assigned three sequential addresses in upper memory space making it appear as a memory device to the CPU. Two addresses are used for the Address Register which holds the address to be trapped. The third address is a register within the Control Unit that enables and disables the trap mechanism.

The Comparator compares the contents of the Address Register with the Address Bus. When the two are equal, the Comparator signals the Control Unit via EQUAL. If the trap is enabled, a Memory Disable signal is generated. This signal is used to turn the memory off so that the Data Bus is free for use by the trap. When the CPU signals its desire to read an instruction from memory, the trap jams its own instruction onto the Data Bus. In the case of an 8080, this instruction is an RST (RESTART) instruction which is a single-byte unconditional CALL.

The routine called by the trap could display the registers, dump memory, enter a new address into the trap, etc. If the trap address is set within a loop, for example, then the results of each pass through the loop could be displayed. Additionally, if input–output devices are placed in memory space, then attempts by the CPU to access these devices may be trapped.

## IV. Address LIFO

There are two things that are certain in the life of a programmer: he will erase a file he shouldn't have and he will write a program that mysteriously branches to never-never land. Therefore, an Address LIFO is proposed as a means of tracing backward through a program to hasten finding the errant code in the solution of the latter problem.

Assume a special stack whose PUSH function was not under direct CPU control but whose POP function was. Each time the CPU references an instruction in memory, the address of the reference is pushed into the stack. At any time the CPU could disable the PUSH operation and examine the elements in the stack. This would permit a programmer to trace the steps of his program back to the depth of the stack. Should his program branch outside of its space, this stack could be examined to see where the program came from.

A block diagram of an Address LIFO is illustrated in Fig. 3. As with the Address Trap, it is assigned sequential addresses in upper memory. A Control Unit determines that the CPU is referencing an address within its allowed address space and pushes that address into the stack. A flip–flop within the Control Unit enables and disables the PUSH operation.

When the CPU desires to access the stack, it commands the Control Unit to cease the Push operation. It can then POP the stack without pushing the stack access routine addresses into the stack.

The LIFO may be used in conjunction with the trap described in Section III. Among other things, the trap routine could fetch the contents of the stack and display it for the programmer.

## V. A Microprocessor Virtual Machine System

Before going into the proposed microprocessor virtual machine system architecture, a brief review of virtual machines will be offered. The interested reader should consult Refs. 1–7 for details.

A virtual machine (VM) is defined as an efficient, isolated copy of a real machine. This concept can be

explained by the virtual machine monitor (VMM) shown in Fig. 4. The VMM is a program that has the following characteristics:

(1) It provides an environment for other programs that is essentially identical to the real hardware environment of the original machine.

(2) Programs executing in this "virtual" environment suffer only small decreases in execution speed.

(3) The VMM exerts complete control over the system resources.

A virtual machine can be thought of as the environment created by the VMM.

A typical form of a VMM and VM implementation is to define a dual-state architecture. Two distinct modes of system operation are defined, privileged and nonprivileged, in which all critical functions are performed in the privileged state. The VMM operates in the privileged mode and performs such functions as direct handling of interrupts, performing input-output, and changing machine state. Each VM under the control of a VMM performs input-output to virtual devices and has the effects of its interrupts simulated by the VMM.

Instructions that must be executed in a privileged mode are called sensitive instructions. An instruction is control sensitive if it attempts to change the amount of resources available to the processor or affects the processor mode. An instruction is said to be behavior sensitive if its execution depends on a real memory address or the processor mode. All non-sensitive instructions are said to be innocuous.

In order for a machine to be virtualizable, the architecture must be such that when a sensitive instruction is executed in a non-privileged machine state, a trap occurs and the privileged state is entered. Consider Fig. 5 which shows the configuration of Fig. 1 in a slightly modified form. The Address Trap device now contains a Base and Bounds Register. The VMM loads these registers with the base and bounds of the VM it desires to execute. Should the VM attempt to access memory outside of these boundaries, an address trap occurs.

A Data Trap is similar in function to the Address Trap except it traps data rather than addresses. It contains a Content Addressable Memory (CAM), which holds the so-called sensitive instructions. When the Data Bus contains one of these instructions when the CPU is doing an instruction fetch, a trap occurs. The other use for this device is to implement instruction macros.

Upon power-up, the VMM can queue VMs and choose one for execution. Upon a trap or after a given delay, the VMM can suspend operation of one VM and start another. Although the details of the machine state switching have not been worked out, a little thought should prove them to be tractable.

## References


1. Buzen, J. P., and Gagliardi, I. O., "The Evolution of Virtual Machine Architecture," *Proc. NCC*, AFIPS Press, Montvale, New Jersey, pp. 291–300, 1973.

2. Gagliardi, I. O., and Goldberg, R. P., "Virtualizable Architectures" *Proc. ACM AICA International Computing Symposium*, Venice, Italy, 1972.

3. Galley, S. W., "PDP-10 Virtual Machines," *Proc. ACM SIGARCH-SIGOPS Workshop on Virtual Computer Systems*, Cambridge, Massachusetts, 1969.

4. Goldberg, R. P., *Virtual Machine System*, Report No. MS-2686, MIT Lincoln Laboratory, Lexington, Mass., 1969.

5. Goldberg, R. P., "Hardware Requirements for Virtual Machine Systems," *Proc. Hawaii International Conference on Systems Sciences*, Honolulu, Hawaii, 1971.

6. Goldberg, R. R., "Architecture of Virtual Machines," *Proc. NCC*, AFIPS Press, Montvale, New Jersey, pp. 309–318, 1973.

7. Popek, G. J., and Goldberg, R. P., "Formal Requirements for Virtualizable Third Generation Architectures," *CACM*, Vol. 17, No. 7, July 1974.

8. *PDP-11 UNIBUS Interface Manual*, Digital Equipment Corporation, Maynard, Massachusetts, April 1970.

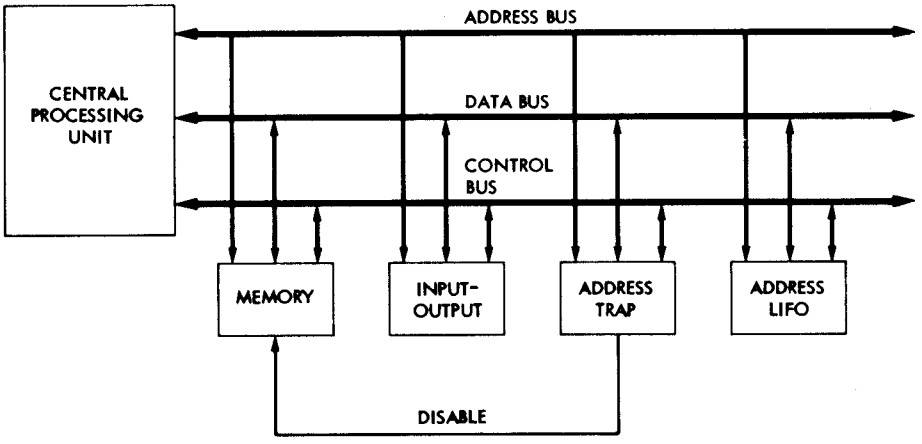


Fig. 1. Basic microprocessor architecture

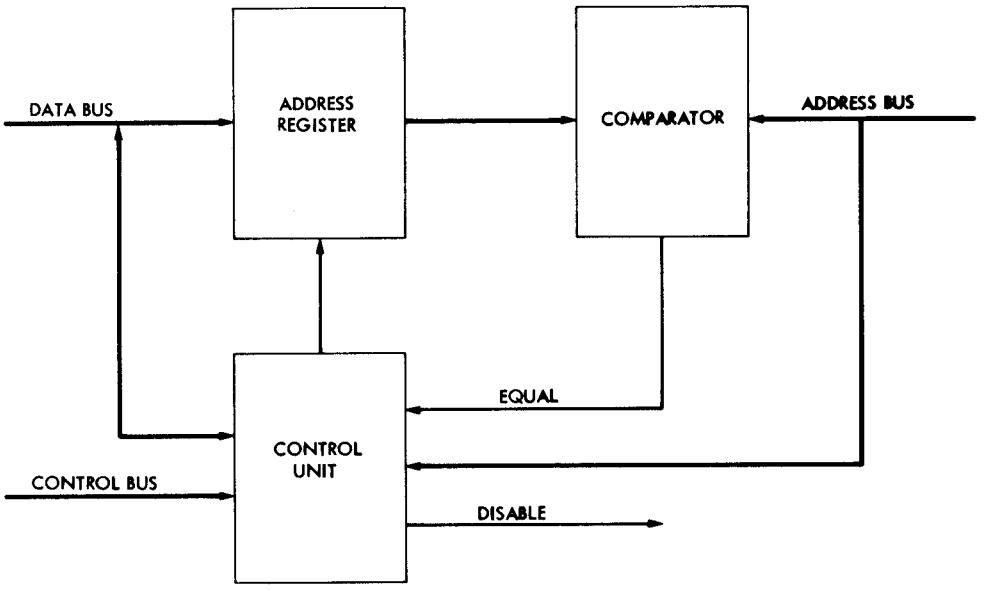


Fig. 2. Block diagram of Address Trap

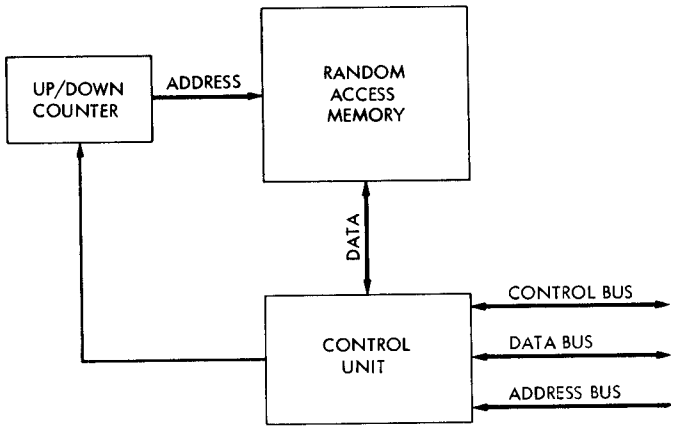


Fig. 3. Address LIFO

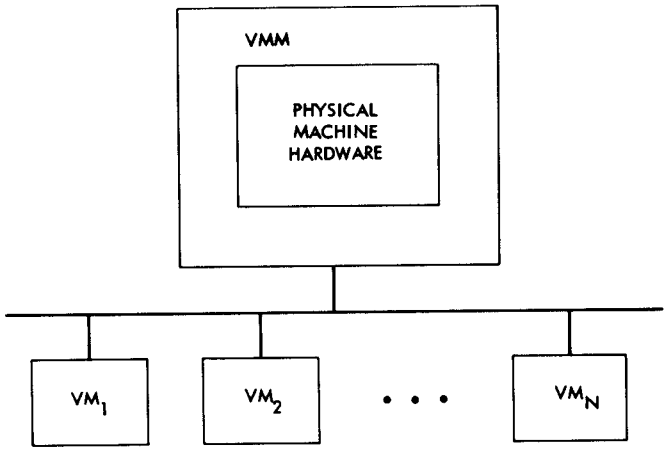


Fig. 4. Virtual machine monitor

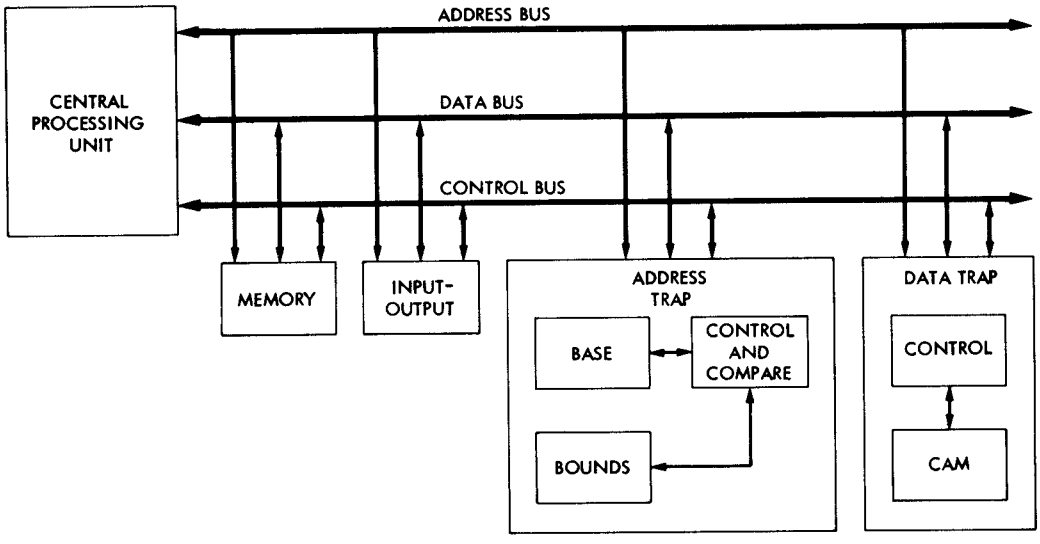


Fig. 5. Microprocessor based virtual machine system architecture