Microprocessor Based Product Design on an HP3000

a Presentation for the
7th International Meeting of the
HP General Systems Users Group
Denver Colorado, November 1978

by
Jack C. Armstrong
Los Altos Research Center

Hardly a day passes without the announcement of a new product or new application based on micro-processors. Concurrent with these announcements are constantly reducing prices for microprocessor hardware. Unfortunately, as applications become more complex, the cost of software development has soared.[1]

Microprocessor manufacturers were quick to respond to complaints about the high cost of developing products around their devices, by introducing microprocessor development systems.[2] These systems, which were valuable in debugging both software and hardware, were immediately placed into wide-spread use. Not unreasonably, microprocessor vendors offered development systems tailored very specifically to their own products, which as a not-coincidental side effect locked their customers into using their chips to the inclusion of other vendors. This effect increased as the cost of the development systems increased. Recently, a few development systems have been introduced (not by microprocessor vendors!) which accommodate several manufacturers lines, but these systems often lack features available in the vendor's custom systems.[3]

Another response to the need for software development aids was the development of so-called "cross software", which allowed the use of a larger, or at least different, computer to compose and assemble/compile programs for microprocessors.[4] This software was made available on all major commercial time sharing services, and organizations with in-house computer systems purchased cross assemblers and cross compilers for use on their own machines. The trend towards cross software has been accelerated by the rapid acceptance of higher level programming languages with the need for larger compilers. The use of another machine for software development has much to recommend it, including freedom to choose from many different microprocessors at minimal incremental expense, use of a common, and usually better, text-editor, etc. The snag in this process comes when the software is written and compiled without syntax errors...now comes actual debugging of the program logic. This, in addition to check-out of the prototype hardware,

generally requires transferring the software to a microprocessor development system, with its capabilities for in circuit emulation, hardware break-points, single stepping thru instructions, etc.

Using the cross software approach frequently means a cost savings in the purchase of development systems, both because fewer of them will be necessary, and the less sophisticated models (without text editors, assemblers, compilers, etc.) may be used. However, it is rare that development systems of some sort will not be required.

One company which found themselves involved in developing a product around a microprocessor chose to stand back and look at the entire process of developing the product, and all that that entailed.[5] This was Scientific Micro Systems, of Mountain View, California. The following steps were identified in the complete project of developing, testing, and supporting a new microprocessor based product:

1. System specification and functional descriptions.

2. Detail design specifications.

3. System implementation, (hardware and software), including construction of prototypes, coding software, etc.

4. System test and evaluation.

5. System documentation, including technical reference manuals, user's guides, manufacturing instructions, sales literature, etc.

6. Product support and maintenance.

The system which evolved at SMS utilized an HP3000, several microprocessor development systems, and a prom burner, all integrated in one system for the development of a microprocessor based product.[6] The total system involved the use of a sophisticated text editor and word processing system (LARC EDITOR/SCRIBE) which included features designed to aid program development, documentation, and tracking changes in both source programs and associated literature.[7] Cross assemblers and cross compilers were implemented on the same system, which produced object code files on the system disc. These files were then "down loaded" onto either the microprocessor development system (for testing) or onto a prom burner to fuse proms for the completed system.

As consultants to SMS, we developed two programs, one for interfacing to the development system, and the other to interface directly to a prom burner. These devices were plugged into HP3000 RS232 terminal ports, and interface programs written in SPL. Once developed, the interface programs were not terribly complex, but the thought and design took considerable effort.

Two approaches were considered in attaching the development system to the HP3000. The first would be to connect an RS232 port on the development system to a terminal port on the HP3000. This would require software in the development system which would allow a user on the development system console to talk "through" the development system to the HP3000, which would then be tricked into thinking that the development system is a user terminal. A user could then log onto the HP3000 and run software to access the object files created by the cross software and transmit then back to the development system. At SMS, the development system being used had its entire operating system in firmware, so no modification was possible. Instead, we chose to fool the development system rather than the HP3000. This was done by connecting the development system console line to a dedicated terminal port on the HP3000. Users may utilize any terminal to log onto the HP3000, and then run an interface program which reads the terminal, looks for a few special commands, and otherwise sends any line typed directly to the development system. The special commands cause disc files (of object code) to be transmitted as though they were being entered by hand directly into the development system. Thus, the development system thinks it is talking to a user console, not an HP3000. (A user who always types at 240 characters per second!).

In other, more flexible development systems, either technique could be used, and each have some advantages. In the case of the prom burner, we again made the device think it was attached to a terminal, not a computer. This program is actually more complex than the interface to the development system, as the prom burner may be both read or written, and we incorporated a facility to compare two blocks of object code, compute check sums, fill unused addresses with a user selectable constant, etc. Also, the program must deal with proms of several different sizes and types.

The success of this installation led to another company acquiring an HP3000 for similar purposes. Caere Corporation, also of Mountain View California, interfaced not only development systems, but single board computers (SBCs) to terminal ports on their HP3000. The SBCs actually have very simple operating systems in firmware which allow down loading via an RS232 port, so programs on the HP3000 can load and run test programs on prototype devices, which then transmit back test results for later analysis on the larger machine. They are using the same text editor and word processing system as SMS, and for similar purposes. They also use cross software, but have a rather unique twist to their use of a development system. One high level microprocessor language currently has a more efficient compiler implemented as a resident compiler on the development system than those available as cross compilers on larger systems. Caere intends to use the HP3000 for text editing and compostion of source programs, but will then ship the source to the development system for compilation!. The resulting object code will then be transferred back to the HP3000's disc, for subsequent down loading onto SBC's connected to other ports. Thus, the development system

becomes a "compiling box", and rarely if ever will be used for anything else.

The advantages of directly interfacing to microprocessor devices, via standard RS232 interfaces, are that each component of the complete system is doing whatever it is best at, with a maximum of flexibility for future changes. The major improvement in this scheme would be to upgrade the cross compilers available. Ideally, one should be able to chose from several high level languages to develop software with, and then select the optimum microprocessor for the application. This selection would be considerably aided by a choice of code emitters all working off the same compiler, so code density and instruction counts could be compared. Finally, a direct interface to a development system would allow final testing of the competed system. The HP3000 has proven to be an ideal nucleus for such a total system, and most of the software is available today.

---

[1] McDermott, Jim, "Experts Tell How to Hold Down the High Cost of Microprocessor Programs", Electronic Design 26, vol. 23, (Dec. 20, 1975) pp20-26.

[2] Bursky, Dave, "It's Getting Easier to Program Microprocessors With the New Software Design Aids", Electronic Design 2, vol. 25, (Jan. 18, 1977) pp20-26.

[3] Snigier, Paul, "Microprocessor Development Systems - Which One is 'Best'", EDN, vol. 27, no. 5, (Mar. 5, 1977) pp68-78.

[4] Nauful, Eli S., "Software Support for Microprocessors Poses New Design Choices", Computer Design, vol. 15, no. 10, (Oct. 1976) pp93-98.

[5] Ivie, Evan L., "The Programmer's Workbench -- A Machine for Software Development", Communications of the ACM vol. 20, no. 10, (Oct. 1977) pp746-753.

[6] Auclair, Dan, "A Minicomputer-Based Microprocessor Development System", Compcon 78 Spring, 16th IEEE Computer Society International Conference, pp334-337.

[7] Rochkind, M. J., "The Source Code Control System", IEEE Transactions on Software Engineering, SE-1, 4 (Dec. 1975) pp364-369.