p51-56.

The IBM 5100 Portable Computer Design.

A. J. Cubbon
955/002-1
GSD-Rochester
4-456-6271

Abstract

The IBM 5100 Portable Computer is an example of a new approach to product development for IBM, and of a new level of miniaturization for computers. Rather than design all the components of the 5100, the development team used existing components wherever possible. This method extended to the language level, as illustrated by the APL implementation, and is responsible for a 50% savings of development time. The new Field Effect Transistor Read Only Memory (FET ROM) with 48K bits per chip allowed up to 246K bytes of storage in the desk top unit. This permitted implementing APL and BASIC and made large program areas possible.

IBM 5100

INTRODUCTION

In September 1975, IBM entered the small, desk top computer marketplace with the 5100 Portable Computer. This computer was made available worldwide in April 1976. The 5100 represents a new approach for IBM. From a very small development team to its dedicated sales force, the techniques used were new to IBM. One of the most significant breaks from the IBM tradition was the integration approach to development. Instead of having a large development group designing all the 5100 components, a relatively small team developed most of the 5100 from existing IBM components. The implementation of the APL programming language is a prime example of this new development approach.

The 5100 also represents a new level of miniaturization of electronic components. The new 48K bits per chip Field Effect Transistor Read Only Memory (FET ROM) permitted memory capacities previously unavailable in a desk top computer. As you may know, the 5100 has a maximum capacity of 248K of memory. Without this size memory, the 5100 could not have accommodated APL and BASIC on the same machine or allowed the 64K storage needed for the more complex user programs.

5100 DEVELOPMENT

The 5100 started as an idea of an IBM engineer. He and another IBM engineer developed the idea and presented it to personnel at IBM's General Systems Division Headquarters in Atlanta, Georgia. This resulted in the design of a prototype. The prototype supported the APL language and used an audio-type cassette tape as its permanent storage media. The APL used on the prototype was originally implemented on the IBM 1130 Computer. An 1130 emulator was written in the native microlanguage of the prototype. The success of the prototype led to the creation of a 5100 development team in late 1973.

Two significant differences in the prototype design were introduced at the outset of development. First, the BASIC language was added because of its popularity as an interactive computer language in the United States. Second, the storage media was changed from the audio-type cassette tape to the new data cartridge tape which offered improved reliability, and greater speed and capacity. Although not part of the original prototype, a hard copy unit and communications capabilities were always in the plan. The hard copy unit selected was the matrix printer already used on the IBM System/32 and the 3740 Data Entry System because it was fast, reliable, and available. The last change made in the design plan was to go from the two 1130 implementations of APL already examined to the version used on the IBM System/370, known as APLSV.

The only hardware specially designed for the 5100 is the power supply and the covers. The keyboard is a standard IBM solid state keyboard used in System/32 and the 3740. The read/write memory is used in IBM System/370 and System/32. Other than improved density, the Read Only Memory (ROM) is the same technology as used in the 3740. Most of the other components were developed for other IBM products and used on the 5100 without modification.

Because of this large number of existing components used in the 5100, the development team only designed interface logic and wrote code. This code was written in microcode for device support and emulators or the higher-level code that was being emulated.

Emulators were required because of the three-level architecture used in the 5100 for language support. Rather than writing the language interpreters directly in microcode, a more eificient use of memory was possible if an intermediate language was used.

The original plan was to use the most popular version of the 1130 APL implementation that was used in the prototype. This had the advantage of being able to use the 1130 emulator from the prototype. Because of the limited memory on the 1130, an overlay technique was used in which pages of code, about 150 instructions each, were brought in from disk when the routine in that page was to be executed. This is similar to the hardware virtual storage concept. The plan was to bring the pages from the dense, 48K bit ROM instead of disk.

Later, this version had been rejected and replaced with another 1130 APL. The new version contained many more of the new functions of APL. This version--which also used an overlay technique with page sizes up to 300 instructions--was studied for several months. Later, it was rejected because the overlay routines required too much time and the overlay pages cut too deeply into the available user storage. It also required modifications for improving the numeric precision and for adding the latest APL System Variables and System Functions.

At this time, one of the team members introduced the idea of using the IBM System/370 version of APL called APLSV. This version had a significant number of advantages over the 1130 APLs. It was the latest version with all the latest APL language features and functions. It had 15 digits of numerical precision vs six in the 1130 versions. It had an extensive input/output subsystem which was not available in the early APL systems. And most significantly, it was the most widely used and tested version of APL in IBM. The one drawback was the lack of a 370 emulator for which there was only 12K executable ROM left.

In late June 1974, the development team undertook the formidable task of writing an emulator and the necessary higher-level code to interface the 370 APL interpreter with the 5100 keyboard display, tape and printer. To date, the finished emulator has remained error free and uses less than 10K of ROM.

There were three major interface routines:

a. A supervisor that interfaced to the keyboard and displayed like a terminal. It also permitted using the printe as a hard copy of the display.

b. A command processor that supported the APL program library commands like SAVE and LOAD for the tape cartridge.

c. An input/output processor that allows the APL programmer to use the tape and printer as independent devices unde program controls.

The initial work was completed by December 1974. Through three people's effort, the APL system was running in only six months; a task that normally takes considerably more time. Fina testing and ROM manufacture was completed in less than a year. Because we had a known system to test against, the emulator was so effectively debugged no errors have been found since early 1975 when final testing was completed.

In addition to greatly reducing the time and manpower requirements, the technique of using already existing code yielded a cleaner and a more compatible product. The only differences between the 5100 version of APL and APL on the System/370 were the portions dependent on the input/output devices, display vs typewriter, and tape vs direct access devices.

5100 ARCHITECTURE

There are three types of storage in the 5100: read/write and two ROMs. All of these memories are Metal On Silicon Field Effect Transistor (MOS FET) Large Scale Integration (LSI) chips. The read/write is the same as that used in the System/32 and some models of the System/370. It serves a dual purpose as registers for the microprocessor and emulators, and storage for user programs.

The newest ROM is the dense, 48K bits per chip storage used for the APL and BASIC interpreters. This storage contains the intermediate language instructions. In the case of APL, these are System/370-like instructions. This storage is accessed by the emulators for each intermediate language as if it was a

direct access storage device rather than like memory. The other ROM is not as dense, but is high speed allowing microinstructions to be executed directly out of it. This storage is used for the two emulators, the input/output device support code and the input/output supervisor.

Storage represents the majority of the electronics in the 5100. The remaining electronics form the microprocessor and input/output device support cards. There is a display card, an I/O control card and external device cards.

The nonexecutable ROM is also attached through the I/O adapter. This causes it to be accessed like an I/O device using a GET microinstruction rather than through a storage address register. The data and instructions in read/write memory and executable ROM are fetched using a storage address register in the microprocessor.

The microprocessor is a fast two-chip 8-bit processor. It has 49 16-bit microinstructions and can operate on four interrupt levels. The levels are prioritized in this order: 1) Keyboard; 2) Tape and Printer; 3) Communications and Serial I/O; 4) Normal Operations. The display operates using cycle steals to fetch two characters from the display buffer in memory.

The microprocessor can fetch instructions from either executable ROM or read/write storage via a 16-bit (plus 2 bits of parity) data path. This allows features like Serial I/O and Communications to have their supporting microcode in read/write memory. It also makes possible the patch capability which allows changes to the logic flow by superceding portions of ROM instructions with instructions in read/write storage.

The interrupt levels allow each set of I/O adapters to use the microprocessor on a demand basis. In the low addresses of read/write memory, there are four sets of sixteen 16-bit registers, one for each interrupt level. Unless masked, control is given to the highest level interrupt pending at the end of executing the current microinstruction.

The display also has a 16-bit (plus 2 bits of parity) data path from read/write storage to the display logic. The display operates in two modes. In the NORMAL mode, data is fetched via cycle steals from the display buffer and displayed as character data. In REGISTERS mode, the data is fetched from the microprocessor register area, converted to hexadecimal (base 16, 0-9, A-F) and displayed for diagnostic use.

The input/output adapter is accessed by using microinstructions. It has an 8-bit (plus 1 parity bit) data path. This smaller data path was used because of the relatively lower performance requirements. This adapter is not only used to access the nonexecutable ROM, but all the other I/O devices.

The two 5100 features not yet mentioned are the Communications Feature and the Serial I/O Adapter Feature. The Communications Feature causes the 5100 to function like a standard asynchronous terminal to a remote system. However, it has local editing support and the tape cartridge can be used to store results from or send input to the remote system. This makes it superior to a simple typewriter terminal. Results stored on the data cartridge can subsequently be used locally by APL or BASIC programs on the 5100.

The Serial I/O Adapter Feature allows the user to attach many peripheral devices to the 5100 such as plotters, display screens, card readers, and printers. This feature allows a standard R.232C serial channel, like a tape unit. The characteristics of this channel can be configured to suit the device being attached. The user can alter such parameters as line rate, parity, and code, which provides a great range in the devices that can be attached.

CONCLUSIONS

The IBM 5100 has contributed two developments to the data processing industry. It has achieved a new level of storage provided in a desk top, portable computer and it has demonstrated that the technique of using existing components can be extended from hardware to programming with the same kind of development time savings.

REFERENCES

"IBM 5100 Maintenance Information Manual," SY31-0405-2, IBM General Systems Division, 1976.