IBM MINI A RADICAL DEPARTURE

New system architecture produces "remarkable" price/performance benefits.

An experimental IBM minicomputer called the "801," the product of a four-year development effort at the T.J. Watson Research Center in Yorktown Heights, N.Y., has drawn major attention from IBM's product line divisions with "remarkable" price/performance benefits due to a radically new system architecture.

Dr. Joel Birnbaum, director of the computer science department at the Watson lab, said the 801 development program was the largest on-going project at Yorktown and "one of the most ambitious projects we have ever undertaken." Birnbaum gave a first, albeit sketchy, description of the 801 project at a Brown Univ. symposium that marked the recent inauguration of a full-fledged computer science department at the Providence, R.I., college.

The Watson Research Center is the headquarters for IBM's Research Division, a group separate and distinct from both IBM's product divisions and the IBM R&D Division. The Research Division is generally regarded as IBM's group for blue sky, high risk projects—heavily dedicated to basic research and the realm of possible but improbable potential. Set up in opposition to IBM's overtly product-oriented R&D, the Research Division explores radical alternatives, said Birnbaum—including some that are born, like the 801, with the idea of putting aside all the 360 and 370 architecture. "Starting from scratch," smiled Birnbaum. "Doing it right."

Although Birnbaum and other Yorktown staff cautiously avoid offering any specific performance data, Birnbaum at one point told his Brown audience that the 801, because of its extraordinary speed, will be used as an ultrasensitive "hybrid monitor" for both hardware and software on 370/168 configurations in Yorktown's experimental systems laboratory. In this application, he said, the 801—"executing three or four instructions in the space of a memory cycle of a 168"—could actually "snoop around" in the main memory of a 168 without slowing down the big cpu's performance under heavy burden, to provide previously unattainable performance data.

While considerably less than descriptive, as a hint this is rather exciting. The main memory cycle time on the Mod 3 168 is 320 nanoseconds. If the 801 is three times as fast, it would have a MIPS rate of over 10. As a measurement of performance MIPS rates are suspect, but it is startling to note that the internal speed of the 168 itself has it executing 2.4 million instructions per second (MIPS), and the 3033 has a MIPS rate of 5, according to IBM.

"If it does what it is projected to do, what the simulations indicate it should do, it will give us some fairly significant cost performance advantages over other

The 801 team is only now debugging the first prototype.

machines IBM has made in the past," acknowledged Birnbaum. The 801 team is only now debugging the first prototype, he cautioned, and despite the apparent results of simulation studies, without programs running through the iron "it's still hard to predict how successful it will be." Nevertheless, he said, the Research Division has decided already to build three or four prototypes for internal tests in varied environments.

At Yorktown, 801 project manager George Radin, one of the primary authors of the PL/I language, said the new minicomputer design has drawn unusual enthusiasm from the technical staff. "Good results always lead to good feelings," said Radin, "and from the beginning we've had surprisingly good results."

"We've got quite a few well-known people working on that project, people who have had successes before," said Birnbaum, "so it has attracted attention within the company's product divisions."

The 801 design appears to trade off some of the unused generality of conventional machine architecture—relying more on the support of an intelligent compiler and a high level language—to allow single-cycle execution of hardwired instructions which in conventional systems are interpreted through microcode and executed in several machine cycles. A unique dual cache separates data from instructions and thus allows the 801 to access both in a single cycle.

The 801 concept proposal is credited to IBM fellow John Cocke, who guided much of the architectural and compiler design over the last several years. Birnbaum described it as not so much a single idea as an elegant restatement of observations made over many years of IBM trace tape analysis: studies that contrasted patterns of actual machine use with the generality of function built into the machine primitives.

Trace tape analysis has long highlighted the fact that a relatively small number of a computer's instructions—usually the simplest (store, load, shift, simple branches, etc.)—get vastly more use than the many others. Yet system architecture generally required that even these instructions be interpreted through microcode instruction sets (which require several machine cycles to execute) in order to support a rich and changeable internal interface.

"The question we asked was, 'Isn't this rather silly in terms of the way people use machines?' " said Birnbaum. "Why not take this primitive set of instructions and hardwire them into the machine so that it executes them in a single cycle—and not make believe that we haven't been working with optimizers and compilers for 15 years or so!"

The follow-through involved an unusual multidisciplinary design team—one-third engineers, one-third compiler people, and one-third system programmers doing the OS and file system—who were brought together under Radin to work on the design from the onset. The joint design approach drew some of Yorktown's best talent, said Radin.

"We got together a bunch of programmers and engineers—men with a lot of experience with IBM trace tapes; men who know what our programs do on our machines—and we tried to understand what things are expensive or slow. What in the hardware can an intelligent compiler discover need not be there at all at run time? Or whose function can be done in some degenerate, fast or cheap way at run time?

"And it turns out," said Radin, "there are really many, many such cases. So, we developed a system in which the hardware and software sort of fit together more rationally. You get less of this overgeneralized execution and therefore you get better performance and less cost."

Birnbaum, despite his managerial reticence, bridles at the suggestion that the 801 design loses true generality. "We don't sacrifice that, we just implement it differently," he said. "The question is, do you then pay a performance penalty for implementing those things which were

"Good results always lead to good feelings and from the beginning we've had surprisingly good results."

left out of the hardwire, the things we implement as subroutines of these primitive instructions?

"The question is, how much do you lose? And the answer *seems* to be, although I'm not sure yet, that not only don't you lose—in many cases you actually gain!"

The "almost cross-cultural" design team had several obvious opportunities to reimplement major functions once the architecture was open to challenge. Grafted functions could be integrated. High speed buffered memories and caches are now common, but in the IBM 370s, they were retrofitted into the machine and therefore made transparent to the software. "That seemed to us unnecessary when compilers and operating systems could take very great advantage of the information they could gather by analyzing what the program is doing with stored information," noted Birnbaum wryly.

"The combination of compiler and machine tries to move to compile time whatever can be moved there—because clearly whatever is done in compile time need only be done once, rather than upon every execution of the program," explained Radin. "For those things which *have* to be done at run time," the design team sought the hardware/software combination that would minimize the number of machine cycles, if they could not bring it into a single cycle.

The team adapted a subset of PL/I for the 801 design—PL.8—although Radin foresees no problem in putting FORTRAN and Pascal front-ends onto the compiler. With the complementary design of the 801 and PL.8, however, the user gets some additional security and program protection. "It's possible to do an analysis of what data a program touches and what data a program changes. Possible to do flow analysis on the program in compile time so that in fact you can guarantee that in this program—when it accesses X of I, for instance—I is going to be within the range of the declaration of that array. And, therefore, you don't have to check and see whether by some error it was too big and will clobber some other part of the program."

To limit the cycle requirements—indeed to make possible single cycle execution of instructions—the 801 has dual caches to separate instructions from data. The fetching and storing of data and the fetching and storing of instructions, are thus asynchronous and parallel so that the 801 can, in a single cycle, fetch instructions from the instruction cache, execute the instruction it's already got, and load or store data from the data cache. The dual cache structure is very difficult to build into any conventional architecture, explained Radin. "This was a lot more than just an implementation decision."

"The combination of compiler and machine tries to move to compile time whatever can be moved there."

The 801 basically has two I/O buses; one a direct memory access high speed bus, a standard direct memory access, except it is designed to have high band width; and the other, a program input/output bus which will be used for control and low speed devices.

The 801 I/O structure is also highly unconventional. "We view I/O as being handled through adapters custom designed for the device being connected," explained Birnbaum. "We think those adapters are cheap and easy to do because we think that what has been done previously in a controller can (in the 801) be done in the central processor. We think we have enough performance to make that an efficient way to operate."

—Vin McLellan