Computers

Altering computer architecture is way to raise throughput, suggest IBM researchers

Despite its decreasing cost, hardware still contributes heavily to a computer system's costs. This fact, coupled with an increasing insight into computer software, has led researchers at IBM's Thomas J. Watson Research Center, Yorktown Heights, N.Y., to try to revise computer architectures by discovering new ways of trading off hardware and software.

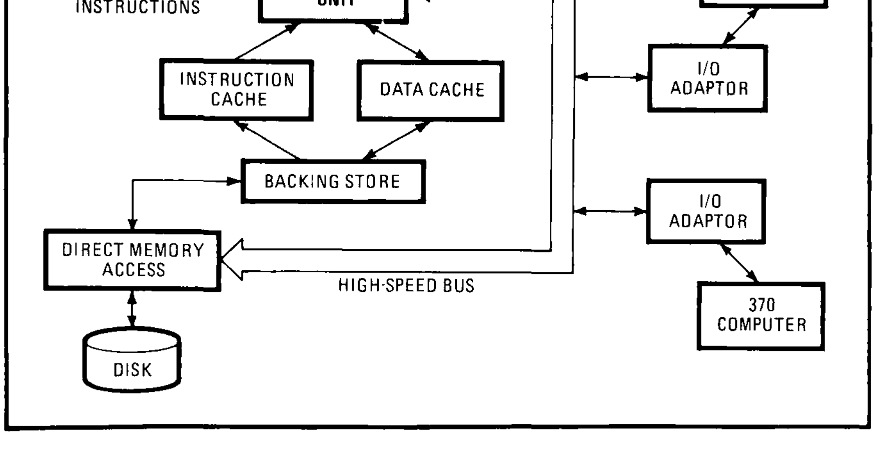
In a project called the 801 Minicomputer, based on the ideas of IBM fellow John Cocke, a group of hardware and software specialists has developed ways to perform frequently executed instructions in hardwired logic, rather than by working from stored microprograms. In addition, they are developing compiler software that scans source-program instructions and then rearranges them for optimum throughput.

No registers. Since most programs consist of frequent executions of only a small number of "primitive" instructions, such as LOAD, STORE, BRANCH, and ADD, the group has designed a central processing unit that executes such instructions directly with its emitter-coupled-logic hardware. The machine has no general-purpose registers. Instead, circuits are specialized—tailored for each instruction. Complex functions, such as floating-point operations, are performed by the primitive instructions as microinstructions.

Going hand in hand with this approach is a development of a new "intelligent" compiler that adjusts the sequence of operations for the optimum throughput. Project manager George Radin points out that today's computers simply interpret instructions one at a time and are "surprised" every time they receive a new instruction, since they have no memory of previous instructions they have performed. However, an intelligent compiler can automatically study the program and rearrange it to reduce redundant operations and the time the CPU may be waiting for new data or instructions.

The group also is studying functions that can be better handled in software. An example Radin cites is memory protection. The hardware needed to decode addresses to prevent unauthorized users from gaining access to certain parts of memory is extensive, he says. But such address checking could be easily handled by the intelligent compiler that compares addresses with user codes.

Looking back, Radin points out



Special mini. Experimental IBM minicomputer uses two cache memories—one for instructions and one for data—each accessible by the central processor. The CPU is made of hardwired logic for direct execution of frequently used instructions.

that the industry's very first compiler, the Fortran I, written for the IBM 704, was an intelligent compiler that was designed to reduce machine running time, which then was more costly than programing time. This balance soon shifted, however, as hardware costs dropped and programing costs increased.

Subsequent compilers concentrated on easing programing by offering English-language commands. Thus, Radin notes, the present project harks back to the early days of compilers in its attempt to perform more tasks with software.

Another key feature of the computer is the use of separate cache memories for instructions and data. Other computers may use a single cache, in which data and instructions are mixed. With separate caches, the central processor can extract information from one cache while the other is busy fetching new information from main memory. However, one question still open, Radin says, is whether the hit ratios—how often the cache already has the requested information, compared to how often it must go out to main memory for it—will be better for the two caches than the hit ratio of a single cache.

Although the group calls the present system a minicomputer, the techniques developed through the study apply to computers of any size, Radin points out. He also points out that the concepts apply to central processors built with any semiconductor technology. □