Many engineers regard the control unit as one of a computer system's most complex components. In the early days of computers, this complexity posed special computer design problems. Because of the inevitable lack of structure and resulting ad-hoc circuitry, such control units became known as random logic.

As an alternative to random logic, Maurice Wilkes of Cambridge University proposed a systematic technique to design and implement control units in 1951. This technique represented the control unit's overall signal sequence issued to the computer as a rather primitive instructional set encoded in a memory array. Noting the obvious programming analogy, Wilkes called his technique microprogramming, and the instructional sequence became known as a microprogram. Resulting microprogram structure regularity avoided much of random logic's arbitrariness, thereby greatly facilitating computer design. By implementing the EDSAC-2 computer, Wilkes and his Cambridge colleagues demonstrated the practicality of their approach.

In applied science, the lag between a new concept's birth and its commercial realization can often be considerable. Although several subsequent developments occurred in Europe and America, microprogramming's first large-scale application was IBM's System/360 computer series in the early 1960's. In the two decades since, microprogramming has become widely used for control unit implementation.

While most microprogrammed computers prior to 1970 held the microprogram in read-only memories, storage technology advances led to writable-control-store availability. Writable control stores enable microprogrammers to overwrite microprogram store contents—in principle, users can change a computer's
behavior by changing writable-control-store contents. As a result, microprogramming attracted not only computer designers but also system and even application programmers. The notion of migrating operating systems and software functions into microcode, thereby enhancing system performance and system security, has been seriously investigated since the mid-1970's. Coined as a synonym for microprogram, the term "firmware" emphasizes the position of microprograms relative to programs (software) and circuits (hardware).

Finding tools to aid and ease the writing, testing, and implementation of microprograms has concerned microprogrammers from the start. Since increased use of microprogramming has led to progressively larger microprograms, this concern eventually resulted in a firmware engineering discipline—loosely defined as establishing the scientific principles underlying microprogramming and applying these principles to develop and implement firmware.¹

Rather than attempting a summary overview, we will present reports on some interesting problems under study—this special issue features six current firmware engineering research projects. We chose IEEE Software as our forum because firmware engineering involves transferring theory, principle, and technique from software to firmware, with appropriate adaptations. We might fruitfully apply resulting technical and theoretical knowledge back to software; for example, we might use microcode compaction techniques to optimize compilers for array/signal/systolic processors, and we might apply firmware verification principles to verify real-time programs. By highlighting firmware engineering research—and results—we hope to provide a base for further knowledge flow across software/firmware boundaries.

Subrata Dasgupta is the USL Foundation professor of computer science at the University of Southwestern Louisiana. His current research interests span the areas of computer architecture, hardware description languages, and design methodology. He has taught at Ohio State University and the University of Alberta, and was a senior visitor at the University of Cambridge Computer Laboratory. During the summer of 1986, he will be a visiting fellow at Wolfson College (Oxford) and the University of Oxford Computation Laboratory.

He served as program chairman for the 16th annual ACM-IEEE microprogramming workshop in 1983, and is a member of the IFIP working group 10.2 (digital system description and design tools). He is the author of The Design and Description of Computer Architectures (Wiley-Interscience, 1984), and is presently writing a book entitled Computer Architecture: A Modern Synthesis to be published by John Wiley and Sons.

Dasgupta received his BE from the University of Calcutta in 1967, and his MS and PhD in computer science from the University of Alberta in 1974 and 1976. He is a member of the ACM and IEEE Computer Society. His address is the Center for Advanced Computer Studies, University of Southwestern Louisiana, PO Box 44330, Lafayette, LA 70504-4330.

Robert A. Mueller is associate professor of computer science and director for research at Colorado State University's George T. Abell Center for Computer-Assisted Engineering. He is also a member of the National Technological University computer science and computer engineering faculties. His current research interests include retargetable compilers, microcode generation and optimization, logic programming, and automated reasoning.

Mueller received his BS in computer science and his MS in mechanical engineering from Colorado State University in 1974 and 1976, and his PhD in computer science from the University of Colorado in 1980. He was a visiting scholar at the Technische Hochschule in Aachen, Germany during 1985. His address is the Computer Science Department, 604 South Howes Street, Colorado State University, Fort Collins, CO 80523.

References