International Journal of Parallel Programming, Vol. 25, No. 2, 1997

From the Guest Editors

It is with great pleasure that we introduce this two-part special issue on Instruction-Level Parallel Processing. With the entire computer industry continuing to turn out hardware and software products based on instruction-level parallel processing technology, this is a particularly timely special issue on the subject area.

The papers that appear in this special issue were chosen based on the recommendations of the Program Committee of the *28th Annual ACM/IEEE International Symposium on Microarchitecture*. Six outstanding papers from the symposium were selected. The authors were invited to revise the papers to meet journal quality standards. All of the authors responded with enthusiasm and hard work, and the result is the distinguished collection of six papers presented in this special issue.

In Part I Volume 25, Number 2 the first paper, "Efficient Instruction Scheduling Using Finite State Automata" by Bala and Rubin, extends a technique for detecting pipeline resource hazards based on finite state as automata. These extensions support efficient instruction scheduling across basic blocks in the presence of complex hardware resource constraints. The extensions include a space requirement reduction technique, a technique to allow a basic block to inherit resource consumption from multiple predecessors, and a dual automata technique to support scheduling across basic blocks.

The second paper, "Optimization of VLIW Compatibility Systems Employing Dynamic Rescheduling," by Conte and Sathaye, presents a technique called Dynamic Rescheduling that supports binary compatibility across generations of VLIW processors, by creating hardware-specific VLIW code on demand, at each page fault. The authors show experimental results based on the IMPACT compiler and the TINKER architectural framework, and propose a caching scheme for further performance improvements.

The third paper, "Region-based Compilation: Introduction, Motivation, and Initial Experience," by Hank, Hwu, and Rau, introduces a new compilation approach where the compiler synthesizes fragments of code

0885-7458/97/0400-0051$12.50/0

828/25/2-1

called "regions" as compilation units, in place of the traditional procedure-based compilation paradigm. Regions can include frequently executed code that crosses procedure boundaries. Region-based compilation allows the compiler to control problem size and complexity while exposing inter-procedural scheduling and optimization opportunities. The authors show initial experimental results based on the IMPACT compiler.

In Part II Volume 25, Number 3 the first paper, "Techniques for Critical Path Reduction of Scalar Programs," by Schlansker and Kathail, describes a family of compiler techniques for branch intensive code, called critical path reduction techniques, which reduce the lengths of critical paths through control and data dependences. The authors illustrate the use of critical path reduction techniques in the context of both superblocks and more general single-entry acyclic regions in scalar code. A new predication model called Fully Resolved Predicates is shown to unify the treatment of control and data dependences in the context of critical path reduction.

The second paper, "The M-Machine Multicomputer," by Fillo, Keckler, Dally, Carter, Chang, Gurevich, and Lee presents an experimental multi-processor system that attempts to maximize both single thread ILP performance and overall system throughput. The M-Machine brings together Instruction Level Parallelism and multithreading hardware, as well as multiprocessor parallelism obtained through efficient message passing. The authors give a detailed description of the machine microarchitecture of each node, as well as the overall system organization for fast synchronization and communication between nodes.

The third paper, "Managing Data Caches Using Selective Cache Line Replacement," by Tyson, Farrens, Matthews, and Pleszkun, presents a microarchitecture technique to characterize the behavior of individual load instructions to support intelligent decisions on cache line replacement. This approach is compatible with existing instruction set architectures. Experimental results show that substantial performance improvement can be achieved with the proposed scheme.

We would especially like to thank all the authors for their extensive efforts to incorporate the (sometimes substantial) modifications suggested by the reviewers. It is the teamwork of the authors and the reviewers that makes this special issue a particularly worthwhile contribution to the field of instruction level parallel processing.

Guest Editors
Kemal Ebcioglu, Thomas J. Watson Research Center, IBM
Wen-mei Hwu, University of Illinois, Urbana-Champaign