Abstract

The design of many-core-on-a-chip has allowed renewed an intense interest in parallel computing. On implementation part, it has been seen that most of applications are not able to use enough parallelism in parallel register sharing architecture. The exploitation of potential performance of superscalar processors has shown that processor is fed with sufficient instruction bandwidth. The fetcher and the Instruction Stream Buffer (ISB) are the key elements to achieve this target. Beyond the basic blocks, the instruction stream is not supported by currents ISBs. The split line instruction problem depreciates this situation for x86 processors. With the implementation of Line Weighted Branch Target Buffer (LWBTB), the advance branch information and reassembling of cache lines can be predicted by the ISB. The ISB can fetch some more valid instructions in a cycle through reassembling of original line containing instructions for next basic block. If the cache line size is more than 64 bytes, then there exist good chances to have two basic blocks in the recognized instruction line.

The code generation for parallel register share architecture involves some issues that are not present in sequential code compilation and is inherently complex. To resolve such
issues, a consistency contract between the code and the machine can be defined and a compiler is required to preserve the contract during the transformation of code. In this paper, we present a correctness framework to ensure the protection of the contract and then we use code optimization for verification under parallel code.

Reference

- Marcos, Keali, “Exposing instruction level parallelism in the presence of loops”, Computation Systems Vol. 8 Number 1, pp. 074-085, 2004
- S. Muchnik, “Advanced compiler design and implementation”, Morgan Kaufmann
Index Terms

Computer Science

Parallel Computing

Key words

ILP

Multithreading

Fine-grained

Intthreads

ISB