

v good!

The goal of Instruction Level Parallelism is to improve processor performance by executing multiple instructions simultaneously. One of the factors that limits our ability to execute instructions in parallel are control dependences, such as conditional expressions, which branch to different code segments depending on the value of the expression. Predicated execution is a technique that is used to eliminate control dependences. I researched predicated execution, locating information by searching for the following terms: "predicated execution", "branch" and "predication", and "predication".

Predicated execution conditionally executes an instruction based on the value of one of the instructions source operands, referred to as its predicate [1]. This allows the processor to execute both branches of a conditional expression simultaneously, while only committing one of the branches, thereby "combining both paths of a branch into a single path" [2] and eliminating the control dependence. Support for predication has begun to be included in newer processor architectures, including the Intel Itanium which contains 64 1-bit predicate registers [3]. Additionally, research has been done on compiler techniques which model and minimize a program's control flow in order to facilitate better use of predication [4], as well as on improving program structure in order to exploit Instruction Level Parallelism and predication [5].

While predicated execution can improve performance by reducing the performance loss due to hard-to-predict branches, it can also lead to performance loss in the case of predicated branches that are actually easy-to-predict at runtime [6]. Wish branches is a technique which has been proposed as a solution to this problem. Wish branches "use predicated execution for hard-to-predict dynamic branches, and branch prediction for easy-to-predict dynamic branches, thereby obtaining the best of both worlds" [6].

References

[1] Reference book: R.H. Bishop, ²Ed. *The Mechatronics Handbook*, New York: CRC Press LLC, 2002.

[2] Article from *INSPEC*: L. Carter, B. Simon, B. Calder, L. Carter, and J. Ferrante, "Path analysis and renaming for predicated instruction scheduling," *International Journal of Parallel Programming*, [online], vol.28, {no. 6}, pp. 563-588, {Mar. 2000}.

[3] Book: J.S. Evans and G.L. Trimper, *Itanium Architecture for Programmers: Understanding 64-bit Processors and EPIC Principles*, New Jersey: Prentice Hall, 2003.

[4] Article from *Compendex*: D.I. August, J.W. Sias, J. Piuatti, S.A. Mahlke, D.A. Connors, K.M. Crozier, et. al., "Program decision logic approach to predicated execution," *Conference Proceedings - Annual International Symposium on Computer Architecture, ISCA*, [online], pp. 208-219, {1999}.

[5] Article from *Science Citation Index Expanded*: F. Zhang and E.H. D'Hollander, "Using hammock graphs to structure programs", *IEEE Transactions on Software Engineering*, [online], vol. 30, {no. 4}, pp. 231-245, {Apr. 2004}. (This source cited L. Carter [4])

[6] Article from *IEEE Explore*: K. Hyesoon, O. Mutlu, Y.N. Patt, J. Stark, "Wish branches: enabling adaptive and aggressive predicated execution," *IEEE Micro*, [online], vol. 6, {no. 1}, pp. 48-58, {Jan.-Feb. 2006}.

italicize book journal and conference titles