WCET analysis with locked instruction caches (Lock-MS)

Alba Pedro-Zapater, Juan Segarra, Rubén Gran and Víctor Viñals-Yúfera
Instituto de Investigación en Ingeniería de Aragón (I3A),
Departamento de Informática e Ingeniería de Sistemas,
Universidad de Zaragoza
C/ Mariano Esquillor s/n
Zaragoza, Spain 50018

Clemente Rodríguez
Dpt. Arquitectura y Tecnología de Computadores,
Universidad del País Vasco
Manuel Lardizabal Ibilbidea 1
Donostia, Spain 20018

ABSTRACT

The WCET (worst case execution time) of a program is difficult to compute because of the lack of predictability of conventional caches. In this work we propose a new WCET analysis module which considers a program in its binary form executing in a real-time processor with locked down instruction caches. We assume the instruction selection to be performed by the existing *Lock-MS* method. Our module is able to give the set of ILP constraints needed both for the WCET computation and the optimal selection of the instruction cache lines to be locked into the cache. As a result we can compute the WCET for a large number of system configurations, including cache design parameters. Moreover, our module is able to analyse realistic complex programs with a large number of paths.

ACM Reference format:
Alba Pedro-Zapater, Juan Segarra, Rubén Gran and Víctor Viñals-Yúfera and Clemente Rodríguez. 2017. WCET analysis with locked instruction caches (Lock-MS). In *Proceedings of ACM Celebration of Women in Computing, Barcelona, Spain, 2017,* 1 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Hard real-time systems need to satisfy stringent timing constraints. In general, upper bounds on the execution times are needed to satisfy these constraints. Real-Time systems are increasingly present in the industry and the daily life. We can find examples in many sectors such as avionics, robotics, automotive, manufacturing, or air traffic control. A real-time system consists of a number of tasks that can be organized by priorities and they have to be scheduled in a way so that they satisfy their deadlines. In order to guarantee their correctness, worst-case execution time (WCET) and schedulability have to be analysed. Given hardware components with a fixed latency, the WCET of a single task could be calculated from the partial WCET of each basic block of the task. However, in order to improve performance, current processors perform many operations with a variable duration. These operations come from components such as caches, pipelines, branch prediction, and other speculative components [4]. One of the main challenges in the WCET analysis is the analysis of the memory hierarchy [2]. Cache behaviour depends on past references and, in general, it is necessary to know the previous access sequence in order to predict the latency of a given access.

To avoid the difficulty to predict the behaviour of conventional caches in a precise way, locked down caches may be used. Locked down caches disable the replacement mechanism and contents are stored previously to task execution. As the content is known and it does not change, the misses and hits computation is trivial and it does not depend on the access sequence.

The challenge of locking cache lines is to determine the best set to lock, knowing that locality exploitation may be limited because the selected cache lines will stay during the whole task execution. In order to guarantee safe upper bounds for the WCET, static analysis methods have been used successfully (e.g. [3]). These methods are based in the analysis of the program operations and the control flow graph, either from the source code or the binary. However, one of the main disadvantages of static analysis at present is the lack of automatic tools. In this work we show how to implement a WCET analysis module from a binary representation of a program. The module has been implemented for Otawa, an open static analysis tool which considers hardware models to provide a tight overestimation of the WCET [1]. WCET analysis is done by the Lock-MS method [3]. Lock-MS is a WCET analysis method to locked down caches that produces ILP (*Integer Linear Programming*) constraints whose solution in an ILP system not only determines the WCET, but also the right cache content that minimizes it. Our module is able to generate the ILP constraints defined by the method and this automation allows to analyse the WCET of a large number of experiments (cache parameters, compiler optimizations, etc.) in a simple and productive way. Moreover our module is able to analyse complex programs. Key contributions of this work are the following:

- Automatic WCET analysis and optimization with locked instruction caches (previous work required manual tuning).
- Automatic generation of a compact Lock-MS model [3], whose solving time is linear with respect to the number of conditionals in the task (previous Lock-MS models had an exponential solving time).
- WCET analysis of large benchmarks.

REFERENCES

[1] C. Ballabriga et al. 2010. OTAWA: An Open Toolbox for Adaptive WCET Analysis. In *Proceedings of the 8th IFIP WG 10.2 Inter. Conference on Software Tech. for Embed. and Ubiq. Syst. (SEUS'10).* Springer-Verlag, Berlin, Heidelberg, 35–46.

[2] L. C. Aparicio et al. 2008. Avoiding the WCET Overestimation on LRU Instruction Cache. In *Proc. 14th IEEE Inter. Conference on Embed. and Real-Time Computing Syst. and Apps. (RTCSA 08).* IEEE Computer Society Press, Kaohsiung, Taiwan, 393–398. https://doi.org/10.1109/RTCSA.2008.10

[3] L. C. Aparicio et al. 2011. Improving the WCET computation in the presence of a lockable instruction cache in multitasking real-time systems. *Journal of Systems Architecture* 7 (Aug. 2011), 695–706.

[4] Wilhelm et al. 2008. The Worst-case Execution-time Problem – Overview of Methods and Survey of Tools. *ACM Trans. Embed. Comput. Syst.* 7, 3 (May 2008), 36:1–36:53. https://doi.org/10.1145/1347375.1347389

ACM Celebration of Women in Computing, 2017, Barcelona, Spain
2017. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn