TACLe STSM: Refined WCET analysis for a stack-based data cache

Report

Alexander Jordan

February 19, 2015

Summary

This TACLe STSM visit at the U2IS department at ENSTA ParisTech, allowed us to continue our work on static analysis techniques for a stack-based data cache (stack cache), which has been implemented in hardware for the timing-predictable Patmos processor [ABS13]. The following summarizes the activities that were conducted as part of the STSM:

- Implementation of an abstract-interpretation-based stack cache analysis (based on AbsInt's aiT).
- DAC 2015 manuscript submission: Implementation of a Software Stack Cache and its Static Analysis (analysis as above, based on AbsInt's aiT).
- Principal work for an integrated stack cache analysis in the OTAWA WCET analysis framework.
- Further academic activities including seminar presentations given at ENSTA and CEA, and demonstrating the Patmos WCET toolchain to new collaborators at ENSTA.

Main work and results

Abstract-interpretation-based stack cache analysis

Instead of a separated analysis as in [JBS13] the internal state of a hardware stack cache can be tracked using an approach based on abstract interpretation or data-flow analysis. AbsInt's aiT WCET analysis through its AIS annotation language, allows us to augment its value analysis (abstract interpretation) phase to keep track of the

stack cache state at relevant program points. These points and thus the number of required annotations are limited to stack cache *reserve* instructions (at the start of a function) and *ensure* instructions (typically after a call to a (sub-)function). Based on the derived cache states at all annotated sites, aiT can account for penalty cycles for the cache overhead to the global WCET bound.

Our first evaluations based on the Debie1 benchmark problems from the 2011 edition of the WCET tool challenge[1], show that performing stack cache analysis in this way does not add any significant overhead, while cache costs are accurately bounded. We still expect that a separated stack cache analysis ([JBS13]) can benefit larger applications (with complex calling behavior), but based on the properties of real-time benchmark programs currently available to us, the value-analysis approach is sufficient and at the same time easier to maintain.

Our aiT-based stack cache analysis is available as part of the Platin[2] tool, which drives WCET analysis (using aiT as a timing backend) for Patmos.

Software stack-cache

We submitted a manuscript to the Design Automation Conference (DAC) 2015, which describes the implementation of a stack cache without hardware support, i.e., without special instructions as part of the instruction set, but using (local) scratch-pad memory. In the same paper we discuss the static analysis aspects of a software stack cache, which are based on our experience with aiT. In this case, the analysis using aiT is almost out-of-the-box, meaning that *reserve* and *ensure* operations no longer need to be annotated, though some user annotations are still required. These need to ensure that value analysis, which is inherently conservative in aiT, does not unnecessarily invalidate the values of the processor registers, which contain the pointers into the stack cache region and thus the stack cache state.

OTAWA

Another integrated approach to stack cache analysis can be achieved through an ILP-based model, which integrates with the IPET phase of WCET analysis. The user annotations supported by AbsInt's aiT analysis are however not powerful enough to augment the IPET ILP for our purposes. We thus chose OTAWA as an open-source WCET analysis framework, which at the start of the STSM, already provided limited architectural support for our target, the Patmos processor. However, before being able to perform stack cache analysis using OTAWA, we had to work toward accurately modeling the special instructions in the Patmos instruction set and their cache behavior. More work is needed in that regard. During the STSM we created a Patmos branch of OTAWA including our changes, which is easier to build and maintain (available online[3]).

[1] `http://www.mrtc.mdh.se/projects/WCC/2011/`
[2] https://github.com/t-crest/patmos-llvm
[3] https://github.com/alexjordan/otawa

Further academic activities

The STSM furthermore offered the opportunity to present the results of the T-CREST project, i.e. the Patmos architecture and its WCET-centric compiler framework, in seminar talks given at ENSTA and as part of a real-time systems study group at CEA.

Working in the U2IS department also enabled me to partly supervise Amine Naji, who recently started working towards his PhD, during his initial efforts with WCET analysis and the Patmos toolchain.

Towards the end of the STSM visit, I was able to attend the Ninth Meeting of the French Compilation Community[4], which was organized by the U2IS department and held in Paris.

Future activities

Future work based on the outcome of this STSM visit will focus on OTAWA's WCET analysis capabilities for the Patmos processor. This is planned as a shared effort between the Patmos team at DTU, Amine Naji and Florian Brandner at ENSTA, with involvement of Hugues Cassé at IRIT, Toulouse, who maintains OTAWA.

References

[ABS13] Sahar Abbaspour, Florian Brandner, and Martin Schoeberl. A Time-predictable Stack Cache. In *Proceedings of the Workshop on Software Technologies for Embedded and Ubiquitous Systems*, SEUS '13, 2013.

[JBS13] Alexander Jordan, Florian Brandner, and Martin Schoeberl. Static Analysis of Worst-case Stack Cache Behavior. In *Proceedings of the 21st International Conference on Real-Time Networks and Systems*, RTNS '13, pages 55–64. ACM Press, 2013.

[4] `http://compilfr.ens-lyon.fr/neuvieme-rencontre-compilation/`