



AMAS-BT 2008 CALL FOR PAPERS

1st Workshop on Architectural and Microarchitectural Support for Binary Translation
Held in conjunction with the 35th Int'l Symposium on Computer Architecture (ISCA-35)

Beijing, China -- June 21, 2008

<http://amas-bt.cs.virginia.edu/>

Workshop Organizers

- Mauricio Breternitz, Intel
- Robert Cohn, Intel
- Youfeng Wu, Intel

Program Committee

- Erik Altman, IBM
- Mauricio Breternitz, Intel
- Mark Charney, Intel
- Robert Cohn, Intel
- Andy Glew, Intel
- Kim Hazelwood, University of Virginia
- David Kaeli, Northeastern University
- Chris J. Newburn, Intel
- Alex Skaletsky, Intel
- Chenggang Wu, CAS, China
- Youfeng Wu, Intel

Important Dates

- Abstract due: May 4, 2008
- Submission: May 11, 2008
- Notification: Late May 2008

How to Submit

Please send email to:

mauricio.breternitz.jr@intel.com

- plain text 200 word abstract, authors, title, contact email, by May 4, 2008
- publication-ready submission of less than 5000 words in IEEE style, 2-column, 10-point .doc, .pdf, or .ps format, by May 11, 2008

Long employed by industry, large scale use of Binary translation and on-the-fly code generation is becoming pervasive both as an enabler for virtualization, processor migration and also as processor implementation technology. The emergence and expected growth of just-in-time compilation, virtualization and Web 2.0 scripting languages brings to the forefront a need for efficient execution of this class of applications. The availability of multiple execution threads brings new challenges and opportunities, as existing binaries need to be transformed to benefit from multiple processors, and extra processing resources enable continuous optimizations and translation.

The main goal of this half-day workshop is to bring together researchers and practitioners with the aim of stimulating the exchange of ideas and experiences on the potential and limits of **Architectural and MicroArchitectural Support for Binary Translation**. The key focus is on challenges and opportunities for such assistance and opening new avenues of research. A secondary goal is to enable dissemination of hitherto unpublished techniques from commercial projects.

The workshop scope includes support for decoding/translation, support for execution optimization and runtime support. It will set a high scientific standard for such experiments, and requires insightful analysis to justify all conclusions. The workshop will favor submissions that provide meaningful insights, and identify underlying root causes for the failure or success of the investigated technique. Acceptable work must thoroughly investigate and communicate why the proposed technique performs as the results indicate.

Copies of papers presented at the workshop will be made available at the workshop, and in archival form on the Web.

To better publicize this work, papers from this workshop are candidates for a special publication of "The Best of ISCA 2008 Workshops", likely a special issue of IJPP, the International Journal on Parallel Programming.

Submission Topics

Hardware assistance for translation and code discovery:

- Interpretation engines, decoding assistance, translated code dispatch
- On-the-fly reconstruction of CFGs and data dependences, scheduling and optimization
- Bug-per-bug compatibility issues
- Static translation: without runtime assistance/translation and with runtime assistance/translation (Hybrid Translation)

Hardware assistance for runtime management

- Self-modifying code, self-referential code, precise exceptions
- Runtime information: profiling branch directions, causes of cache misses, memory access monitoring

- Management of translated code and adapting code to changing program behavior, persistent translation, incremental translation
- Multi-many cores: parallel translation, auto parallelization, speculative execution

Hardware assistance for optimization:

- Extra/enhanced internal/physical registers
- Speculative execution support
- Reduced footprint/low-power cores enabled by binary translation, area and power efficiency
- Techniques for parallelizing single-thread programs