

## Method for achieving high hit rate for an address translation cache in binary translation

*M. K. Gschwind*

Disclosed is a method for increasing the cache hit rate of a branch address translation cache by significantly reducing the number of cache misses due to function return. The described approach is based on preloading function return address translations to reduce the number of cache misses and employed in conjunction with the previously disclosed instruction address translation cache. While instruction address caches have shown significant overall performance, their hit rate for function returns has shown a significant amount of cache misses for easily predictable mappings.

In previous work, a return address stack for keeping address translations for function call/return mappings has been proposed to solve this problem, but this comes at the cost of additional hardware and processor state which has to be maintained. In addition, programs which do not always follow the function call/function return path strictly, such as C programs using the longjmp mechanism or C++/Java programs which use the throw/catch exception mechanism, may incur significant performance penalties by disturbing the return stack mechanism. Furthermore, special handling is required for cases where function return does not take the expected path. (Typically, in this case a regular translation lookup has to be performed.)

The present invention increases branch address translation cache hit rate for function return address translation with a unified instruction address translation mechanism, which eliminates the cost of maintaining both a translation cache and a dedicated function return stack. The presented approach gives similar performance to a combined address translation cache with return stack, but only requires a single hardware resource and is also more robust in the presence of unexpected function return behavior since strict function return order is not required.

The present disclosure is based on the branch address translation cache disclosed in [1], and adds an improved cache management mechanism to increase cache hit rate. This management mechanism is based on preloading translations with a high access probability in the near future. The cache is preferably preloaded from problem state to minimize the cost of preloading.

According to the present invention, the following methods are used to implement the various branch instructions:

- Register-indirect branches are performed following an address translation lookup in the cache. If a cache miss occurs, a software handler reloads the translation from a full representation of the table.
- Cross-translation unit calls are executed similarly to register indirect calls.

- On function calls, an address translation tuple <emulated return address, translated return address> is added to the table before control is passed to the called function. This insertion is preferably performed by unprivileged, problem-state program code to minimize overhead.
- On function return, a register indirect branch is performed after translating the architected emulated return address using the cache. This lookup does not usually incur any miss, since the appropriate entry has been recently added to the cache upon the corresponding function call<sup>1</sup> and is most likely still present in the cache. In the unusual case of an address lookup failure, the cache miss handler reloads the translation from a full representation of the table.

The present scheme eliminates all cold start cache misses for function return address translations by using a particular problem-state management approach of the cache. This eliminates all cold start cache misses for function return, and also eliminates many conflict and capacity misses by appropriate reloading (when an entry has already been discarded since its previous use) or by refreshing time usage time stamps (this reduces the likelihood of a cache entry being evicted from the cache in n-way associative caches, when the entry was already present in the translation cache).

In summary, high cache hit rates are achieved by preloading address translations for function return when a function is invoked. The present approach requires no additional resources or access methods over the originally described translation cache, but shows significantly improved performance over the previously disclosed design when simulated using typical scientific and commercial workloads.

While this approach has described branch address translation cache preloading in terms of preloading function return addresses at function call time, other accesses with high access probability may be similarly preloaded at convenient locations following the spirit of this invention.

## References

[1] M. Gschwind, Method and Apparatus for Determining Branch Addresses in Programs Generated by Binary Translation, Research Disclosures, Vol. 41, No. 416.

---

<sup>1</sup> or its time stamp has been updated when such translation was already in the cache.