Delete redundant prefetch instructions generated by loop optimization

Toshihiro KONDA

Background

- **Software Prefetch Instruction**
  - **It is common to use LSU resources**
- **Let's consider unrolling or software pipelining a loop containing a Prefetch instructions.**
- **As a result, there are many cases where the Prefetch instruction performs redundant prefetching.**
- **How can I reduce redundant look-ahead?**

Premise

- There is a method of inserting a prefetch instruction at MVE, but here we consider the simple optimization when outputting a Prefetch instruction to continuous access before a software pipelining.

Prefetch instruction

- Since it is continuous access, one Prefetch instruction,
  - Read data of read address to cache line width.
  - When MVE is performed
    - Prefetch covers the prefetching target and the size is known
    - Obvious from the nature of the Prefetch instruction
  - Then, the following slide equation is obtained.

Equation

- {Cache line width(Byte)} ÷ {Memory access width covered by one Prefetch(Byte)} ÷<truncate> {Unroll number} + 1 = Required Prefetch number

- The number of Prefetch copied by MVE is not optimal, and the number of Prefetch thinned out to the above number is the optimum number of output.

Points to consider

- This method is not strictly optimal.
  - Why?
  - Since Prefetch is thinned out, MRT will have free space in the packed instruction sequence.
  - This is a very simple implementation (although certain effects can be obtained if prefetch instructions use LSU resources).

Conclusion

- I proposed a method to avoid performance degradation as much as possible against Software Prefetch output and loop optimization in classical implementation.

Thanks.