Allocating Rotating Registers by Scheduling
Hongbo Rong  Hyunchul Park  Cheng Wang  Youfeng Wu
Intel Labs

Introduction
A rotating alias register file is a scalable hardware support to detect memory aliases at run-time. This paper solves the problem of allocating it for a software-pipelined loop schedule.

We show that this allocation problem is a software pipelining scheduling problem, and it can be solved fast and efficiently.

Motivating Example
- Unidirectional check: LD specifies only register RR1, but the hardware checks all the higher-indexed registers, including RR1~3.
- Scalability: encode only 1 register to check many.

Rotating alias register file

Unidirectional check

• Unidirectional check: LD specifies only register RR1, but the hardware checks all the higher-indexed registers, including RR1~3.
• Scalability: encode only 1 register to check many.

Algorithm framework

Step 1: Dependence building
Step 2: Modulo scheduling
Step 3: Removing potential false positives
Step 4: Register assignment

An algorithm specially for register allocation
In Step 2, any modulo scheduling algorithm works, but differs in register usage and false positives. We propose a simple algorithm LCP (Local scheduling followed by Pacing):
• Calculate a local scheduling for an iteration
• Pack the schedule of every iteration at RegII
• Heuristics to reduce false positives: Max R, Earliest start

Results

- Solution implemented in Transmeta Code Morphing Software, a dynamic binary translator.
- LCP is compared with JITSP (Just-in-time software pipelining, to appear in CGO’14), RS2 (rotation scheduling), DESP (Decomposed software pipelining), and Ideal (an imagined ideal allocator)
- With 11,825 loops in SPEC2000, LCP shows near-ideal register usage. It is faster than the other algorithms and with less false positives, thanks to the two heuristic.
- On average, LCP has 0.16 false positives per iteration; LCP takes 2.46% translation time, roughly 0.07% of total time.
- Performance impact: measured with 24 hot loops, with static, but without rotating, alias registers, II increases by 11% ~116%, indicating significant performance degradation.

Generalization

An allocation of rotating general registers is also a modulo schedule of lifetimes, where the initiation interval $R = 1$, and there are resource constraints but no dependence constraints.

This formulation can derive the bin-packing approach of Rau et al. 1992, which was shown to be effective.