Trace Cache

Leon Gu
Dipti Motiani

15-740: Computer Architecture, Fall, 2003
10/01/2003
Papers


Superscalar Processors

- Producer - Consumer
- Instruction Level Parallelism
Motivation

- Exploit ILP
- Fetch Bottleneck
  - Instruction cache misses
  - Branch prediction accuracy
  - Branch prediction throughput
  - Noncontiguous instruction fetching
  - Fetch unit latency
A trace is a sequence of instructions starting at any point in a dynamic instruction stream.

It is specified by a start address and the branch outcomes of control transfer instructions.
Fetch Mechanism

- Trace cache is accessed in parallel with instruction cache.
  - Hit ➞ Trace read into issue buffer
  - Miss ➞ Fetch from instruction cache

- Trace cache hit if
  - Fetch address match
  - Branch predictions match

- Trace cache is NOT on the critical path of instruction fetch.
Fetch Mechanism

Instruction Cache
- 1st BB
- 2nd BB
- 3rd BB

Trace Cache
- 1st BB
- 2nd BB
- 3rd BB

Line-Fill Buffer

Instruction Latch

To Instruction Buffers

Fetch Address A

hit?

Take output from trace cache if trace cache hit; otherwise, take output from instruction cache.
Design Issues

- Trace Length
- Sizing
- Indexing
- Branch Throughput

- Fill Mechanism
- Partial Matches
- Associativity
- Replacement Policy
• Present a micro-architecture incorporating a trace cache
  □ Control flow prediction and instruction supply at trace level

• Evaluate performance advantage

• Design issues – size and associativity
Microarchitecture
Microarchitecture

- Trace-level sequencing
- Instruction-level sequencing
- Next trace prediction
- Trace selection
- Hierarchical sequencing
Performance of Fetch Models

- SEQ.1
- SEQ.1-adj
- SEQ.n
- SEQ.n-adj
- TC(64K,4)
- TC-perfect

IPC

go gcc jpeg li perl m88k vortex
Integer Fetch IPC as a Function of Trace Cache Size

Fetch IPC

Trace Cache Entries

- go
- m88ksim
- gcc
- compress
- li
- ijpeg
- perl
- vortex
Fill Mechanism
Power consumption
Filling unit latency
Duplication of instructions
Liveness of traces
Design issues
Paper 2

- Present a **block-based** trace cache implementation
  - Fetch address renaming
  - Basic block cache
- Performance comparison between conventional and block-based trace cache
Motivation

- Trace cache storage efficiency.
- Reduce the latency of indexing and associativity.
- Flexibility of trace construction and prediction.
Comparison

Conventional

Block-based
Questions

- Dependence on branch prediction

- Other mechanisms
  - Branch Address Cache
  - Collapsing Buffer
    - Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching

- Compiler Techniques?
Discussion - Research

- Replace instruction cache with trace cache?
- Reduce duplication and fragmentation
- Dynamic direction prediction trace cache
- Pentium4: Execution trace cache stores 12K decoded micro-ops
Microarchitecture