A Study of Control Independence

Eric Rotenberg
Introduction

• High instruction-level parallelism (ILP)
  - requires large instruction scheduling window
  - branches make this difficult

• Branches introduce control dependences
  - must execute branch to fetch next instructions
  - correct branch prediction/speculation: essentially eliminates control dependences
  - mispredictions are still a major bottleneck: complete squashing
Introduction

• Solutions to branch misprediction bottleneck
  1. better branch prediction
  2. forms of multi-path execution
  3. predication
  4. control independence
Control Independence

- Incorrect control dependent instructions
- Correct control dependent instructions
- Wasted resources
- False data dependences
- True data dependences
- Re-convergent point
- Control independent instructions
Goals

1. Establish new performance bounds of control independence under implementation constraints
   - window size
   - fetch/issue width
2. Identify and quantify primary limiting factors
3. Assess basic implementation requirements and complexity

• Goals 1 and 2 are the focus of this talk
  - Inspired by Lam & Wilson limit study
  - Want to refine understanding of control independence
  - Can aid control independence design
• **WR**
  - Model *Wasted Resources* due to incorrect control dependent instructions
  - Resources => window space and pipeline bandwidth
  - Conversely: nWR => no wasted resources

• **FD**
  - Model *False Dependences* created by incorrect control dependent instructions
  - Conversely: nFD => no false dependences
Models

Endpoints

1. oracle: oracle branch prediction
2. base: conventional prediction/speculation

Control independence models

3. nWR-nFD: no wasted resources, no false dependences
4. nWR-FD: no wasted resources, false dependences
5. WR-nFD: wasted resources, no false dependences
6. WR-FD: wasted resources, false dependences
oracle

FETCH | ISSUE
---|---
1 | 1: r5<=
3 | 3: r4<=
4 | 4: <=r5
| 4: <=r4

**Actual path**

r5 ← 1 → 3 ← 4 ← 2 ← r5

r5 ← 2 ← r4

r5 ← 4 ← r4
nWR-nFD

TIME

FETCH

ISSUE

1: r5 <=
4: <=r5

3: r4 <=
4: <=r4
nWR-FD

FETCH ISSUE

TIME

M

D

1: r5<=
4: <=r5
3: r4<=
4: <=r4

actual path

r5 <=

r5 <=

r4 <=

r4 <=
WR-nFD

- r5 ← 1
- r5 ← 2
- r5 ← 4
- r4 ← 3
- r4 ← 4

Actual path: r4

FETCH

<table>
<thead>
<tr>
<th></th>
<th>ISSUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1: r5&lt;=</td>
</tr>
<tr>
<td>2</td>
<td>4: &lt;=r5</td>
</tr>
<tr>
<td>3</td>
<td>3: r4&lt;=</td>
</tr>
<tr>
<td>4</td>
<td>4: &lt;=r4</td>
</tr>
</tbody>
</table>
WR-FD

M 3 1

D 4 3

r5 ←

FETCH ISSUE

TIME

actual path

1: r5 <=
2: r5 <=
3: r4 <=
4: <=r5
4: <=r4

r5 ←

r4 ←

4 ←r4

<r5
cycle
base

[Diagram with nodes labeled 1, 2, 3, 4 and arrows indicating the actual path.]

FETCH ISSUE

TIME

FETCH | ISSUE
1    | 1: r5<=
2    | 3: r4<=
4    | 4: <=r5
     | 4: <=r4

actual path
Ideal Study

go
oracle
nWR-nFD
nWR-FD
WR-nFD
WR-FD
base
Summary of Study

• Control independence seriously limited by incorrect control dependent instructions
  - wasted window space, fetch/execute resources
  - false data dependences

• Still has large potential
  - potentially reduce gap between oracle and real branch prediction by 1/2
Applications

• Apply insight to guide implementations of WR-FD model
  - e.g. Trace Processors

• Large WR component suggests designs capable of “absorbing” incorrect paths
  - e.g. “expandable window” and multiple fetch units of Multiscalar Processors
  - nWR-FD model => performance potential for Multiscalar with aggressive data dependence resolution/recovery

• The study suggests several new directions...
  - nWR-nFD model
  - Interesting ways of handling data dependent/control independent instructions
Implementation Study

- Basic requirements for misprediction recovery
  - Detect the re-convergent point
  - Insert/remove instructions from middle of window
  - Form correct data dependences
  - Selectively reissue instructions
Implementation Study

Improvement of CI over BASE

- gcc
- go
- comp
- jpeg
- vortex

% IPC improvement

window size

- 128
- 256
- 512
Other Limiting Factors

• Factors exposed by implementation study
  - False mispredictions
  - Incoherent global branch history
  - Re-fetching/Re-dispatching control independent instructions
  - ...and many others. (See Technical Report and revisions.)
Implementation Study Revisited

- no re-fetch
- correct global branch history
- eliminate false misp.
- CI

% improvement over base
Closing Remarks

• Control independence has been gaining momentum
  
  Several limit studies  Multiscalar/Spec MT
  Instruction reuse  Trace processors
  CMU studies  Michigan studies
  DMT  Instruction recycling in SMT

• We’ll see even more control independent designs in the future => misprediction bottleneck

• Now’s the time to understand the primary and secondary factors affecting control independence