Bungee Jumps: Accelerating Indirect Branches Through Hardware/Software Co-Design

Daniel S. McFarlin
Craig Zilles
Indirect Branches Are Increasingly **Predictable**

![Chart showing indirect branch predictability for Nehalem, Sandy Bridge, Haswell, and TAGE.](chart-url)
Indirect Branches Are Increasingly Predictable

And Unbiased

predicted/Kilo Instrs

Mispredicts/Kilo Instrs

Nehalem  Sandy Bridge  Haswell  TAGE

btree  fannkuch  fasta  richards  nqueens  revcomp  float  specnorm  regexdna  knuke  mandelbrot  Geomean

B e t t e r
In-Order Machines Specialize Based on **Branch Bias** or Eliminate **Branch Prediction** Altogether
In-Order Machines Specialize Based on Branch Bias or Eliminate Branch Prediction Altogether

```
if s->type == Circle
  area()
else if s->type == Rect
  area()
else if s->type == Square
  area()
else
  RCPO
```

Shape → VTable

```
30
area() → VTable
60
area() → VTable
10
RCPO
```
In-Order Machines Specialize Based on Branch Bias or Eliminate Branch Prediction Altogether

Shape ➔ VTable ➔ area()

if s->type == Circle
(area())
else if s->type == Rect
(area())
else if s->type == Square
(area())
else
(area())

if (obj is type B);
    r = B::func( );
else if (obj is type C);
    r = C::func( );
else if (obj is type D);
    r = D::func( );
else
    r = obj->func( );

if( ! (p0 | p1 | p2))
    r = obj->func( );
Challenge: Non-Reconvergence & Large Number of Targets
Challenge: Non-Reconvergence & Large Number of Targets

A
ld r9, [rax*8+0x94]
jmp r9

B
ld r8, [rip+0x5b]
ld edi, [rsi*4+0x92]
movsxd rcx, r8
ld r10, [rcx*4+0x92]
cmp edi, r10
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edx, 0x6
jz I

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz J

E
ld edx, [rcx*8+0x10]
test edx, edx
jz K

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
ld r10, [rcx*4+0x46]
cmp edi, r10
jnz M

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
jnz L

sjeng: f_in_check

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
Challenge: Non-Reconvergence & Large Number of Targets
Challenge: Non-Reconvergence & Large Number of Targets
Challenge: Non-Reconvergence & Large Number of Targets

A

\[ \text{ld r9, [rax*8+0x94]} \]
\[ \text{jmp r9} \]

B

\[ \text{ld r8, [rip+0x5b]} \]
\[ \text{ld edi, [rsi*4+0x92]} \]
\[ \text{movsxd rcx, r8} \]
\[ \text{ld r10, [rcx*4+0x92]} \]
\[ \text{cmp edi, r10} \]
\[ \text{jnz H} \]

C

\[ \text{ld edx, [rsi*4+0x7d]} \]
\[ \text{cmp edx, 0x6} \]
\[ \text{jz I} \]

D

\[ \text{ld ecx, [rip+0x58]} \]
\[ \text{ld esi, [rip+0x81]} \]
\[ \text{lea edi, [rsi+rcx*1]} \]
\[ \text{cmp edi, edx} \]
\[ \text{jz J} \]

E

\[ \text{ld edx, [rcx*8+0x10]} \]
\[ \text{test edx, edx} \]
\[ \text{jz K} \]

F

\[ \text{ld r8, [rip+0x39]} \]
\[ \text{ld edi, [rsi*4+0x46]} \]
\[ \text{movsxd rcx, r8} \]
\[ \text{ld r10, [rcx*4+0x46]} \]
\[ \text{cmp edi, r10} \]
\[ \text{jnz M} \]

G

\[ \text{ld r8, [rip+0x5e]} \]
\[ \text{ld edi, [rsi*4+0x42]} \]
\[ \text{movsxd rcx, r8} \]
\[ \text{ld r10, [rcx*4+0x42]} \]
\[ \text{cmp edi, r10} \]
\[ \text{jnz L} \]
Challenge: Non-Reconvergence & Large Number of Targets

A
ld r9, [rax*8+0x94]
jmp r9

B
ld r8, [rip+0x5b]
ld edi, [rsi*4+0x92]
movsxd rcx, r8
ld r10, [rcx*4+0x92]
cmp edi, r10
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edx, 0x6
jz I

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz J

E
ld edx, [rcx*8+0x10]
test edx, edx
jz K

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
ld r10, [rcx*4+0x46]
cmp edi, r10
jnz M

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
jnz L

sjeng: f_in_check

Text
Challenge: Non-Reconvergence & Large Number of Targets
Challenge: Non-Reconvergence & Large Number of Targets

A
ld r9, [rax*8+0x94]
jmp r9

B
ld r8, [rip+0x5b]
ld edi, [rsi*4+0x92]
movsxd rcx, r8
ld r10, [rcx*4+0x92]
cmp edi, r10
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edx, 0x6
jz I

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz J

E
ld edx, [rcx*8+0x10]
test edx, edx
jz K

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
ld r10, [rcx*4+0x46]
cmp edi, r10
inz M

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
inz L

sjeng: f_in_check

Text
Challenge: Non-Reconvergence & Large Number of Targets

A
ld r9, [rax*8+0x94]
jmp r9

B
ld r8, [rip+0x5b]
ld edi, [rsi*4+0x92]
movsxd rcx, r8
ld r10, [rcx*4+0x92]
cmp edi, r10
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edi, 0x6
jz 1

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz J

E
ld edx, [rcx*8+0x10]
test edx, edx
jz K

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
ld r10, [rcx*4+0x46]
cmp edi, r10
inz M

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
inz L

sjeng: f_in_check

Text

1
5
95
4
96
1
99
1
99
25
24
19
14
11
7
Challenge: Non-Reconvergence & Large Number of Targets

A
ld r9, [rax*8+0x94]
jmp r9

B
ld r8, [rip+0x5b]
ld edi, [rsi*4+0x92]
movsxd rcx, r8
ld r10, [rcx*4+0x92]
cmp edi, r10
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edx, 0x6
jz D

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz E

E
ld edx, [rcx*8+0x10]
test edx, edx
jz F

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
ld r10, [rcx*4+0x46]
cmp edi, r10
inz M

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
inz L

sjeng: f_in_check

Text
Challenge: Non-Reconvergence & Large Number of Targets

A
  ld r9, [rax*8+0x94]
  jmp r9

B
  ld r8, [rip+0x5b]
  ld edi, [rsi*4+0x92]
  movsxd rcx, r8
  ld r10, [rcx*4+0x92]
  cmp edi, r10
  jnz H

C
  ld edx, [rsi*4+0x7d]
  cmp edx, 0x6
  jz I

D
  ld ecx, [rip+0x58]
  ld esi, [rip+0x81]
  lea edi, [rsi+rcx*1]
  cmp edi, edx
  jz J

E
  ld edx, [rcx*8+0x10]
  test edx, edx
  jz K

F
  ld r8, [rip+0x39]
  ld edi, [rsi*4+0x46]
  movsxd rcx, r8
  ld r10, [rcx*4+0x46]
  cmp edi, r10
  jnz L

G
  ld r8, [rip+0x5e]
  ld edi, [rsi*4+0x42]
  movsxd rcx, r8
  ld r10, [rcx*4+0x42]
  cmp edi, r10
  jnz M

sjeng: f_in_check

1 5 95

25 24 19 14

7 11

1 99

4 96

99

99

99 99 99
Challenge: Non-Reconvergence & Large Number of Targets

A
ld r9, [rax*8+0x94]
jmp r9

B
ld r8, [rip+0x5b]
ld edi, [rsi*4+0x92]
movsxd rcx, r8
ld r10, [rcx*4+0x92]
cmp edi, r10
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edx, 0x6
jz I

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz J

E
ld edx, [rcx*8+0x10]
test edx, edx
jz K

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
ld r10, [rcx*4+0x46]
cmp edi, r10
inz M

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
inz L

sjeng: f_in_check

Text
Missed Optimization Opportunity: Next Branch Bias
Missed Optimization Opportunity: Next Branch Bias
Missed Optimization Opportunity: Next Branch Bias

Branch Bias

80 85 90 95 100

bench btree fannkuch fasta fastaredux knuke mandelbrot nbod...
Exploiting Predictability: Benefit from Next Branch Bias
Exploiting Predictability: Benefit from Next Branch Bias

A
- `ld r9, [rax*8+0x94]`
- `pred-indirect-jump`

B
- `ld r8, [rip+0x5e]`
- `ld edi, [rsi*4+0x42]`
- `movsxd rcx, r8`
- `ld r10, [rcx*4+0x42]`
- `cmp edi, r10`
- `jnz H`

C
- `ld edx, [rsi*4+0x7d]`
- `assert r9, C`
- `cmp edi, 0x6`
- `jz I`

D
- `ld ecx, [rip+0x58]`
- `ld esi, [rip+0x81]`
- `lea edi, [rsi+rcx*1]`
- `assert r9, D`
- `cmp edi, edx`
- `jz J`

E
- `ld edx, [rcx*8+0x10]`
- `assert r9, E`
- `test edx, edx`
- `jz K`

F
- `ld r8, [rip+0x39]`
- `ld edi, [rsi*4+0x46]`
- `movsxd rcx, r8`
- `assert r9, F`

G
- `ld r8, [rip+0x5e]`
- `ld edi, [rsi*4+0x42]`
- `movsxd rcx, r8`
- `assert r9, G`

H
- `ld r10, [rcx*4+0x42]`
- `cmp edi, r10`
- `jnz M`

I
- `ld r8, [rip+0x5e]`
- `ld edi, [rsi*4+0x42]`
- `movsxd rcx, r8`

J
- `ld r10, [rcx*4+0x42]`
- `cmp edi, r10`
- `jnz L`

K
- `Hoist From`
Exploiting Predictability: Benefit from Next Branch Bias
Exploiting Predictability: Benefit from Next Branch Bias
Exploiting Predictability: Benefit from Next Branch Bias
Exploiting Predictability: Benefit from Next Branch Bias

A
ld r9, [rax*8+0x94]
pred-indirect-jump

B
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
ld r10, [rcx*4+0x42]
cmp edi, r10
assert r9, B
jnz H

C
ld edx, [rsi*4+0x7d]
cmp edx, 0x6
jz I
assert r9, C

D
ld ecx, [rip+0x58]
ld esi, [rip+0x81]
lea edi, [rsi+rcx*1]
cmp edi, edx
jz J
assert r9, D

E
ld edx, [rcx*8+0x10]
cmp edx, edx
jz K
assert r9, E

F
ld r8, [rip+0x39]
ld edi, [rsi*4+0x46]
movsxd rcx, r8
assert r9, F

G
ld r8, [rip+0x5e]
ld edi, [rsi*4+0x42]
movsxd rcx, r8
assert r9, G

H
ld r10, [rcx*4+0x42]
cmp edi, r10
jnz M

I
Hoist From

J
Hoist From

K
Hoist From

L
Hoist From

M
Hoist From

N
Hoist From

O
Hoist From

P
Hoist From
Exploiting Predictability: Benefit from Next Branch Bias
Exploiting Predictability: Benefit from Next Branch Bias
Challenge: Invalid Predicted Target Address

```
0: r0 = load[a]
1: r2 = r0 + r1
2: jmp [r2]
```

Valid Targets: {A, G}
Predict A

Prediction Point
**Challenge: Invalid Predicted Target Address**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>r0 = load[a]</td>
</tr>
<tr>
<td>1:</td>
<td>r2 = r0 + r1</td>
</tr>
<tr>
<td>2:</td>
<td>jmp [r2]</td>
</tr>
</tbody>
</table>

Valid Targets: \{A, G\}

Predict A
Challenge: Invalid Predicted Target Address

Valid Targets: \{A, G\}
Predict A
Challenge: Invalid Predicted Target Address

**Prediction Point**

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: r0 = load[a]</td>
<td>...</td>
</tr>
<tr>
<td>1: r2 = r0 + r1</td>
<td>0: r0 = load[a]</td>
</tr>
<tr>
<td>2: jmp [r2]</td>
<td>predict</td>
</tr>
</tbody>
</table>

Valid Targets: \{A, G\}
Predict A

```
A
B
C
D
```

```
A
B
C
D
1: r2 = r0 + r1
2*: r3 = load[r2]
2: resolve r3, A
```
Challenge: Invalid Predicted Target Address

Valid Targets: \{A, G\}
Predict A

Prediction Point

\[
\begin{array}{l}
0: r0 = \text{load}[a] \\
1: r2 = r0 + r1 \\
2: \text{jmp} \ [r2] \\
\end{array}
\]

\[
\begin{array}{l}
0: r0 = \text{load}[a] \\
\text{predict} \\
\end{array}
\]

\[
\begin{array}{l}
A \\
B \\
C \\
D \\
\end{array}
\]

\[
\begin{array}{l}
A \\
B \\
C \\
D \\
2: \text{resolve} \ r3, A \\
\end{array}
\]

\[
\begin{array}{l}
1: r2 = r0 + r1 \\
2*: r3 = \text{load}[r2] \\
\end{array}
\]
Challenge: Invalid Predicted Target Address

Valid Targets: \{A, G\}
Predict A

Prediction Point
Challenge: Invalid Predicted Target Address

Valid Targets: {A, G}
Predict A

Invalid Target: M
Challenge: Invalid Predicted Target Address

Prediction Point

Valid Targets: \{A, G\}
Predict A

Invalid Target: M

\[
\begin{align*}
0: & \ r0 = \text{load}[a] \\
1: & \ r2 = r0 + r1 \\
2: & \ \text{jmp} [r2]
\end{align*}
\]

\[
\begin{align*}
0: & \ r0 = \text{load}[a] \\
& \text{predict}
\end{align*}
\]

\[
\begin{align*}
0: & \ r0 = \text{load}[a] \\
1: & \ r2 = r0 + r1 \\
2*: & \ r3 = \text{load}[r2] \\
2: & \ \text{resolve} \ r3, A
\end{align*}
\]

\[
\begin{align*}
M \\
N \\
O \\
P
\end{align*}
\]
## Challenge: Invalid Predicted Target Address

### Verifying Valid Targets

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>r0 = load[a]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1:</td>
<td>r2 = r0 + r1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>jmp [r2]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Valid Targets:** \{A, G\}

**Prediction Point:**

1. **Predict A**

2. **Resolve r3, A**

3. **Invalid Target:** M
Solution: Landing Pad

...  
...  
...  

0: r0 = load[a]  

predict 0x0

marker 0x0

A  
B  

1: r2 = r0 + r1  

2*: r3 = load[r2]  

C  
D  

2: resolve r3, A
Solution: Landing Pad

<table>
<thead>
<tr>
<th>marker 0x0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

1: r2 = r0 + r1
2*: r3 = load[r2]
2: resolve r3, A

0: r0 = load[a]

predict 0x0
### Solution: Landing Pad

| ... |
| ... |
| 0: r0 = load[a] |
| **predict 0x0** |
| **marker 0x0** |
| A |
| B |
| 1: r2 = r0 + r1 |
| 2*: r3 = load[r2] |
| C |
| D |
| **2: resolve r3, A** |
Solution: Landing Pad

0: r0 = load[a]
predict 0x0

marker 0x0

A
B
1: r2 = r0 + r1
2*: r3 = load[r2]
C
D
2: resolve r3, A

0: r0 = load[a]
predict 0x0

r2 = r0 + r1
jmp [r2]
Solution: Landing Pad

...  
...  
...  
0: \( r_0 = \text{load}[a] \)

\[\text{predict 0x0}\]

...  
...  
0: \( r_0 = \text{load}[a] \)

\[\text{predict 0x0}\]

\[\text{marker 0x0}\]

A
B

1: \( r_2 = r_0 + r_1 \)

2*: \( r_3 = \text{load}[r_2] \)

2: \( \text{resolve} \ r_3, \ A \)

r_2 = r_0 + r_1

jmp [r_2]
Solution: Landing Pad

0: r0 = load[a]

predict 0x0

marker 0x0

A
B
1: r2 = r0 + r1
2*: r3 = load[r2]
C
D
2: resolve r3, A

0: r0 = load[a]

predict 0x0

r2 = r0 + r1
jmp [r2]
Solution: Landing Pad

0: \( r_0 = \text{load}[a] \)

**predict 0x0**

1: \( r_2 = r_0 + r_1 \)

**marker 0x0**

A

B

1: \( r_2 = r_0 + r_1 \)

2*: \( r_3 = \text{load}[r_2] \)

C

D

2: **resolve** \( r_3, A \)

0: **Prediction Point**

**predict 0x0**

\( r_2 = r_0 + r_1 \)

jmp [r_2]
Solution: Landing Pad

0: r0 = load[a]

predict 0x0

0: r0 = load[a]

predict 0x0

r2 = r0 + r1

jmp [r2]

Invalid Target: M

Fails 0x0 Check

marker 0x0

A

B

1: r2 = r0 + r1

2*: r3 = load[r2]

C

D

2: resolve r3, A

M

N

O

P
Solution: Landing Pad

...  
...  
0: r0 = load[a]  
predict 0x0  
...  
marker 0x0  
A  
B  
1: r2 = r0 + r1  
2*: r3 = load[r2]  
C  
D  
2: resolve r3, A  
...  
...  
0: r0 = load[a]  
predict 0x0  
r2 = r0 + r1  
jmp [r2]  
...  
0: Prediction Point  
r2 = r0 + r1  
jmp [r2]  
...  
0: Prediction Point  
r2 = r0 + r1  
jmp [r2]  
Invalid Target: M  
M  
N  
O  
P  
Fails 0x0 Check
Solution: Landing Pad

0: $r0 = \text{load}[a]$

**marker 0x0**

- A
- B
- 1: $r2 = r0 + r1$
- 2*: $r3 = \text{load}[r2]$
- C
- D
- 2: **resolve** $r3, A$

**predict 0x0**

- $r2 = r0 + r1$
- jmp [r2]

0: Prediction Point

- $r2 = r0 + r1$
- jmp [r2]

Invalid Target: M

Fails 0x0 Check

- M
- N
- O
- P
Solution: Landing Pad

```plaintext
0: r0 = load[a]  predict 0x0

r2 = r0 + r1

cmp [r2]

0: Prediction Point

r2 = r0 + r1

cmp [r2]

Invalid Target: M

Fails 0x0 Check
```

A
B
C
D

1: r2 = r0 + r1
2*: r3 = load[r2]
2: resolve r3, A
Solution: Landing Pad

0: r0 = load[a]

predict 0x0

r2 = r0 + r1

jmp [r2]

marker 0x0

A
B
1: r2 = r0 + r1
2*: r3 = load[r2]
C
D
2: resolve r3, A

Invalid Target: M

0: Prediction Point

predict 0x0

r2 = r0 + r1

jmp [r2]

Fails 0x0 Check

M
N
O
P

Redirect Fetch

No Predict: Stall Fetch
Solution: Landing Pad

... 0: r0 = load[a] predict 0x0

marker 0x0

A
B
1: r2 = r0 + r1
2*: r3 = load[r2]
C
D 2: resolve r3, A

... 0: r0 = load[a]

predict 0x0
r2 = r0 + r1
jmp [r2]

... 0: r0 = load[a]

predict 0x0
r2 = r0 + r1
jmp [r2]

Invalid Target: M

Fails 0x0 Check

M
N
O
P

Redirect Fetch
No Predict: Stall Fetch

r2 = r0 + r1
jmp [r2]
Solution: Landing Pad

Necessitates some changes to indirect call /return handling
Recovery From Misprediction
## Recovery From Misprediction

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>1: r2 = r0 + r1</td>
<td>2*: r3 = load[r2]</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. `resolve fails`
2. `resolve r3, A, RC`
3. `Resteer RC addr to Front End`
4. `Resteer r3 to Front End`
Recovery From Misprediction

1: \[ r2 = r0 + r1 \]
2*: \[ r3 = \text{load}[r2] \]

(1) resolve fails

(2) Resteer RC addr to Front End

(3) Resteer r3 to Front End

2: \text{resolve} \ r3, A, RC
Recovery From Misprediction

(1) resolve fails

(2) Resteer RC addr to Front End

(3) Resteer r3 to Front End

A
B
1: \( r2 = r0 + r1 \)
2*: \( r3 = \text{load}[r2] \)
C
D
2: resolve r3, A, RC
Recovery From Misprediction

A
B
1: r2 = r0 + r1
2*: r3 = load[r2]
C
D

1: resolve fails
2: resolve r3, A, RC
3: Resteer RC addr to Front End
4: Resteer r3 to Front End
Recovery From Misprediction

1. \( r2 = r0 + r1 \)
2. \( r3 = \text{load}[r2] \)
3. Resteer RC addr to Front End
4. Resteer \( r3 \) to Front End
Recovery From Misprediction

A
B
1:  r2 = r0 + r1
2*: r3 = load[r2]
C
D
2: resolve r3, A, RC

(1) resolve fails
(2) Resteer RC addr to Front End
(3) Resteer r3 to Front End
Recovery From Misprediction

1: \( r_2 = r_0 + r_1 \)

2*: \( r_3 = \text{load}[r_2] \)

1: resolve \( r_3, A, \text{RC} \)

(2) Resteer RC addr to Front End

(3) Resteer \( r_3 \) to Front End

(4) Fetch Recovery Code

RC: undo A
undo B
undo C
undo D

(5) \( r_3 \) value from Resteer

(6) Fetch Correct Path starting at \( r_3 \)

r3: E
F
...

...
Recovery From Misprediction

1: \[ r2 = r0 + r1 \]
2*: \[ r3 = \text{load}[r2] \]

(1) resolve fails

(2) Resteer RC addr to Front End

(3) Resteer r3 to Front End

(4) Fetch Recovery Code

RC: undo A
undo B
undo C
undo D

(5) r3 value from Resteer

(6) Fetch Correct Path starting at r3

r3: E
F
...

A
B
C
D
Recovery From Misprediction

1: \( r2 = r0 + r1 \)
2: \( r3 = \text{load}[r2] \)

1: resolve \( r3, A, RC \)
2: resolve \( r3, A, RC \)

2*: \( r3 = \text{load}[r2] \)

(1) resolve fails
(2) Resteer RC addr to Front End
(3) Resteer \( r3 \) to Front End
(4) Fetch Recovery Code
(5) \( r3 \) value from Resteer
(6) Fetch Correct Path starting at \( r3 \)

RC: undo A
undo B
undo C
undo D
\text{jmp} \ r3

r3: E
F
...

...
Recovery From Misprediction

1: \( r2 = r0 + r1 \)

2*: \( r3 = \text{load}[r2] \)

(1) resolve fails

(2) Resteer RC addr to Front End

(3) Resteer r3 to Front End

(4) Fetch Recovery Code

RC: undo A
undo B
undo C
undo D

(5) r3 value from Resteer

(6) Fetch Correct Path starting at r3

r3: E
F
...

10
Recovery From Misprediction

1: \[ r2 = r0 + r1 \]
2*: \[ r3 = \text{load}[r2] \]

(1) resolve fails

(2) Resteer RC addr to Front End

(3) Resteer r3 to Front End

(4) Fetch Recovery Code

(5) r3 value from Resteer

(6) Fetch Correct Path starting at r3

10
Recovery From Misprediction

1: \[ r2 = r0 + r1 \]

2*: \[ r3 = \text{load}[r2] \]

(1) resolve fails

(2) Resteer RC addr to Front End

(3) Resteer r3 to Front End

(4) Fetch Recovery Code

RC: undo A
undo B
undo C
undo D

(5) r3 value from Resteer

(6) Fetch Correct Path starting at r3

r3: E
F
...

...
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

• Maintain correspondence between prediction/resolution
• Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  – Dovetails with existing structures for outstanding branches
• Single pointer, single R/W port
Hardware Requirements

• Maintain correspondence between prediction/resolution
• Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  – Dovetails with existing structures for outstanding branches
• Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

• Maintain correspondence between prediction/resolution
• Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  – Dovetails with existing structures for outstanding branches
• Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

- Maintain correspondence between prediction/resolution
- Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  - Dovetails with existing structures for outstanding branches
- Single pointer, single R/W port
Hardware Requirements

• Maintain correspondence between prediction/resolution
• Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  — Dovetails with existing structures for outstanding branches
• Single pointer, single R/W port
• Maintain correspondence between prediction/resolution
• Small FIFO: not many branches outstanding and no reordering of prediction and resolution instructions
  – Dovetailed with existing structures for outstanding branches
• Single pointer, single R/W port
Experimental Methodology

- Profile Guided Optimization
  LLVM 3.5
- Benchmarks with 0.3% dynamic instruction stream indir branch
- SPEC TRAIN input, PHP and Python first input
- Cycle Accurate x86 simulator PTLSim provides predictability
- Transform non-loop branches with pred > bias and (pred - bias) > 3%
- Run SPEC, PHP, Python on PTLSim using PGO binaries with and without prediction guide on REF
- Speculation support: (alias/shadow registers) for baseline and experimental
- Improved Indirect Branch Predictor: VPC (TAGE study in paper)\textsuperscript{14}

<table>
<thead>
<tr>
<th>Key Structures</th>
<th>Configuration Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bpred</td>
<td>PTLSim default: GShare, 24 KB 3-table direction predictor, 4K-entry BTB, 64-entry RAS</td>
</tr>
<tr>
<td>Front-End</td>
<td>5 stages, Experimentally Varied 2/4 wide Fetch/Decode/Dispatch, 32-entry FetchBuffer</td>
</tr>
<tr>
<td>Execution Ports</td>
<td>Experimentally Varied 2/4</td>
</tr>
<tr>
<td>Functional Units</td>
<td>Up to 2 x LD/ST, 2 x INT/SIMD-Permute, 4 x 64-bit SIMD/FP, 1-cycle bypass</td>
</tr>
<tr>
<td>L1 Caches</td>
<td>8-way 32 KB L1-D$, 4-way 32 KB L1-I$, 64B lines, 4-cycle latency</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>16-way 256KB Unified, 12-cycle latency</td>
</tr>
<tr>
<td>L3 Cache</td>
<td>32-way 4MB LLC, 25-cycle latency</td>
</tr>
<tr>
<td>Miss Handling</td>
<td>64-entry Miss Buffer, 64-entry Load Fill Request Queue</td>
</tr>
<tr>
<td>Main Memory</td>
<td>140-cycle latency</td>
</tr>
</tbody>
</table>
Performance proportional to % of dynamic instructions which are indirect branches (PDS) and amenable to transformation (weighted averaged bias: WAB)

- richards: 4% PDS, 41% WAB
- m88ksim: 0.4% PDS, 57% WAB
- Geomean: 1.1% PDS, 62% WAB
Performance: PHP

- Specnorm: 1.4% PDS, 18% WAB
- run: 0.6% PDS, 49% WAB
- Geomean: 1.4% PDS, 37% WAB
Performance: Python

- **regexdna**: 2.7% **PDS**, 72% **WAB**
- **revcomp**: 1.0% **PDS**, 90% **WAB**
- **Geomean**: 1.8% **PDS**, 73% **WAB**
Conclusions

• Straightforward, low-cost “enabling” transformation
  – Leverages DBT profiling and speculation facilities
• Modest Hardware Requirements
• Leverages Advances In Indirect Branch Prediction
• Good Performance across Integer and Floating Point
• Maintains the Efficiency of the In-Order
Thank You
Thank You

Questions?