BRANCH PREDICTORS

Mahdi Nazm Bojnordi
Assistant Professor
School of Computing
University of Utah
Branch Prediction

- **Goal:** avoiding stall cycles caused by branches
- **Solution:** static or dynamic branch predictor
  - 1. prediction
  - 2. validation and training
  - 3. recovery from misprediction
- **Performance** is influenced by the frequency of branches \((b)\), prediction accuracy \((a)\), and misprediction cost \((c)\)

\[
\text{Speedup} = \frac{\text{Old Time}}{\text{New Time}} = \frac{\text{CPI}_{\text{old}}}{\text{CPI}_{\text{new}}} = \frac{1 + bc}{1 + (1 - a)bc}
\]
A pipelined processor requires 3 stall cycles to compute the outcome of every branch before fetching next instruction; due to perfect forwarding/bypassing, no stall cycles are required for data/structural hazards; every 5th instruction is a branch.

Compute speedup gained by a branch predictor with 90% accuracy

$$Speedup = \frac{(1 + 0.2 \times 3)}{(1 + 0.1 \times 0.2 \times 3)} = 1.5$$
Bimodal Branch Predictors

- One-bit branch predictor
  - Keep track of and use the outcome of last branch

- Shared predictor
- Two mispredictions per loop

Accuracy = 26/30 = 0.86
Bimodal Branch Predictors

- Two-bit branch predictor
  - Increment if taken
  - Decrement if untaken

- A single predictor shared among many branches
- One misprediction on loop exit
- Accuracy = 28/30 = 0.93
- 3-bit predictor?
Using Multiple Counters

- How to assign a branch to each counter?

Program code

... 
branch-1
...
branch-2
...
branch-3

Cost = $n2^a$ bits

1. How many branches are in a program?
2. How many counters are used?
Using Multiple Counters

How to assign a branch to each counter?

- Decode History Table (DHT)
  - Reduced HW with aliasing

Cost = \( n2^b \) bits

Program code

```
... 
branch-1 
...
branch-2 
...
branch-3
```

Least significant bits are used to select a counter

(+): Reduced hardware
(−): Branch aliasing
Using Multiple Counters

- How to assign a branch to each counter?
  - Decode History Table (DHT)
    - Reduced HW with aliasing
  - Branch History Table (BHT)
    - Precisely tracking branches

Most significant bits are used as tags
(+) No aliasing
(−) Missing entries

Cost = \((a-b+n)2^b\) bits

PC

Tags

Counters

hit/miss*
Using Multiple Counters

- How to assign a branch to each counter?
  - Decode History Table (DHT)
    - Reduced HW with aliasing
  - Branch History Table (BHT)
    - Precisely tracking branches
  - Combined BHT and DHT
    - BHT is used on a hit
    - DHT is used/updated on a miss

\[
\text{Cost} = (a-b+2n)2^b \text{ bits}
\]

DHT typically has more entries than BHT
Correlating Branch Predictor

- Executed branches of a program stream may be correlated

```c
while (1) {
  if(x == 0)
    y = 0;
  ...
  if(y == 0)
    x = 1;
}
```

```
while:
  BNEQ R1, R0, skp1
  ADDI  R2, R0, #0
skp1: ...
  BNEQ R2, R0, skp2
  ADDI  R1, R0, #1
skp2: J while
```
Correlating Branch Predictor

- Executed branches of a program stream may be correlated

```
while (1) {
    if(x == 0)
        y = 0;
    ...
    if(y == 0)
        x = 1;
}
```

Global History Register: an r-bit shift register that maintains outcome history.

Cost = $r + 2^{b+r}$ bits
Global Branch Predictor

- GHR is merged with PC bits to choose a counter

Cost = $r + 2^r$ bits
Local Branch Predictor

- One GHR per branch

Cost = $r2^b + n2^x$ bits
Local Branch Predictor

- One GHR per branch

Cost = \( r2^b + n2^b \) bits
Local predictor may work well for some applications, while global predictor works well for some other programs.

Include both and identify/use the best one for each branch.
Branch Prediction Summary

- Dedicated predictor per branch
  - Program counter is used for assigning predictors to branches
- Capturing correlation among branches
  - Shift register is used to track history
- Predicting branch direction is not enough
  - Which instruction to be fetched if taken?
- Storing the target instruction can eliminate fetching
  - Extra hardware is required
Branch Target Buffer

- Store tags and target addresses for each branch