Lecture-9 (Branch Prediction)
CS422-Spring 2018

Biswa@CSE-IITK
Remember This

Sample Program (ISA w/o branch delay slot)

I1: BEQ R4,R3,25
I2: AND R6,R5,R4
I3: SUB R1,R9,R8

Time: t1 t2 t3 t4 t5 t6 t7 t8
Inst
I1: IF  ID  EX MEM WB
I2: IF  ID
I3: IF
I4: I5: I6:

We avoiding stalling by (1) adding a branch delay slot, and (2) adding comparator to ID stage. If we add more early stages, we must stall.

EX stage computes if branch is taken.
If branch is taken, these instructions MUST NOT complete!
Welcome to Branch Prediction

We update the PC based on the outputs of the branch predictor. If it is perfect, pipe stays full!

Dynamic Predictors: a cache of branch history

If we predicted incorrectly, these instructions MUST NOT complete!
If always PC+4?

When a branch resolves
- branch target (Inst_k) is fetched
- all instructions fetched since inst_h (so called “wrong-path” instructions) must be flushed
Flush on a Mispred.

Inst_h: IF_{PC} → ID → ALU → MEM → WB
Inst_i: IF_{PC+4} → ID
Inst_j: IF_{PC+8} → ID
Inst_k: IF_{target} → ID → ALU → WB
Inst_l: IF → ID → ALU → IF

Inst_h is a branch
Branch Prediction

- Idea: Predict the next fetch address (to be used in the next cycle)

- Requires three things to be predicted at fetch stage:
  - Whether the fetched instruction is a branch
  - (Conditional) branch direction
  - Branch target address (if taken)

- Observation: Target address remains the same for a conditional direct branch across dynamic instances
  - Idea: Store the target address from previous instance and access it with the PC
  - Called Branch Target Buffer (BTB) or Branch Target Address Cache
Fetch Stage with BTB and Direction Prediction

Direction predictor (2-bit counters)

Program Counter

Address of the current branch

Cache of Target Addresses (BTB: Branch Target Buffer)

Always taken CPI = \[ 1 + (0.20 \times 0.3) \times 2 \] = 1.12  (70% of branches taken)
Static Branch Prediction

- **Always not-taken**
  - Simple to implement: no need for BTB, no direction prediction
  - Low accuracy: ~30-40%

- **Always taken**
  - No direction prediction
  - Better accuracy: ~60-70%
    - Backward branches (i.e. loop branches) are usually taken

- **Backward taken, forward not taken (BTFN)**
  - Predict backward (loop) branches as taken, others not-taken
Profile-based

- Idea: Compiler determines likely direction for each branch using profile run. Encodes that direction as a hint bit in the branch instruction format.

+ Per branch prediction (more accurate than schemes in previous slide) \(\rightarrow\) accurate if profile is representative!

-- Requires hint bits in the branch instruction format

-- Accuracy depends on dynamic branch behavior:

- TTTTTTTTTTNNNNNNNNNN \(\rightarrow\) 50% accuracy
- TNTNTNTNTNTNTNTNTNTNTN \(\rightarrow\) 50% accuracy

-- Accuracy depends on the representativeness of profile input set
Dynamic Branch Prediction

- **Idea:** Predict branches based on dynamic information (collected at run-time)

- **Advantages**
  - Prediction based on history of the execution of branches
  - It can adapt to dynamic changes in branch behavior
  - No need for static profiling: input set representativeness problem goes away

- **Disadvantages**
  - More complex (requires additional hardware)
Last-Time Predictor

- **Last time predictor**
  - Single bit per branch (stored in BTB)
  - Indicates which direction branch went last time it executed
    
    `TTTTTTTTTNNNNNNNNNN` → 90% accuracy

- Always mispredicts the last iteration and the first iteration of a loop branch
  - Accuracy for a loop with N iterations = (N-2)/N

+ Loop branches for loops with large number of iterations
- Loop branches for loops with small number of iterations

`TNTNTNTNTNTNTNTNTNTN` → 0% accuracy

Last-time predictor CPI = \[ 1 + (0.20\times0.15) \times 2 \] = 1.06 (Assuming 85% accuracy)
Last-Time

predict not taken

actually not taken

predict taken

actually taken

actually not taken

actually taken
Last-time

K bits of branch instruction address

Branch history table of $2^K$ entries, 1 bit per entry

Index

Use this entry to predict this branch:

0: predict not taken
1: predict taken

When branch direction resolved, go back into the table and update entry: 0 if not taken, 1 if taken
Example

Example

```
0xDC08: for(i=0; i < 100000; i++)
    {
        if( (i % 100) == 0 )
            tick();
        if( (i & 1) == 1 )
            odd();
    }
```
Example

DC08: TTTTTTTTTTT ... TTTTTTTTTTTNTTTTTTTTTTTTTTTTTT ... 100,000 iterations

How often is branch outcome != previous outcome? 2 / 100,000

DC44: TTTTT ... TNTTTTT ... TNTTTTT ... 2 / 100

99.998% Prediction Rate

98.0%

DC50: TNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNTNT
Change Predictor after 2 Mistakes

- **pred taken** → **actually taken** → **pred !taken** → **actual !taken** → **pred !taken** → **pred taken**

- "weakly taken" → **pred !taken** → **weakly !taken"" → **pred !taken" → **strongly taken" → **pred taken** → **pred !taken" → **strongly !taken""
Is This Enough

• Control flow instructions (branches) are frequent
  • 15-25% of all instructions

• Problem: Next fetch address after a control-flow instruction is not determined after N cycles in a pipelined processor
  • N cycles: (minimum) branch resolution latency
  • Stalling on a branch wastes instruction processing bandwidth (i.e. reduces IPC)

• How do we keep the pipeline full after a branch?
• Problem: Need to determine the next fetch address when the branch is fetched (to avoid a pipeline bubble)
Is This Enough?

• Assume a pipeline with 20-cycle branch resolution latency

• How long does it take to fetch 100 instructions?
  • Assume 1 out of 5 instructions is a branch
  • 100% accuracy
    • 100 cycles (all instructions fetched on the correct path)
    • No wasted work
  • 99% accuracy
    • 100 (correct path) + 20 (wrong path) = 120 cycles
    • 20% extra instructions fetched
  • 98% accuracy
    • 100 (correct path) + 20 * 2 (wrong path) = 140 cycles
    • 40% extra instructions fetched
  • 95% accuracy
    • 100 (correct path) + 20 * 5 (wrong path) = 200 cycles
    • 100% extra instructions fetched
Who Cares?

- 98% → 99%
  - Who cares?
  - Actually, it’s 2% misprediction rate → 1%
  - That’s a halving of the number of mispredictions
- So what?
  - Halving the miss rate doubles the number of useful instructions that we can try to extract ILP from
  - Piazzaa + 2
Local History & Global History

• Local Behavior
  – What is the predicted direction of Branch A given the outcomes of previous instances of Branch A?

• Global Behavior
  – What is the predicted direction of Branch Z given the outcomes of all* previous branches A, B, ..., X and Y?

* number of previous branches tracked limited by the history length
BTB (What about JALR? Why 30-bit Tag?)

Address of branch instruction
0b0110[...]01001000

Branch instruction
BNEZ R1 Loop

Branch Target Buffer (BTB)

- 30-bit address tag
- target address

Branch History Table (BHT)

4096 entries...

“Hit”

“Taken” Address ↓

“Taken” or “Not Taken”

In real designs, always direct-mapped.

At EX stage, update BTB/BHT, kill instructions, if necessary.

Drawn as fully associative to focus on the essentials.
Two Level Global Branch Prediction [MICRO ‘91]

- First level: **Global branch history register** (N bits)
  - The direction of last N branches
- Second level: **Table of saturating counters for each history entry**
  - The direction the branch took the last time the same history was seen
- Table of saturating counters
What about – NO GHR?

Bimodal predictor: Good for biased branches

(\text{PC} >> 2) \& (2^p - 1)
GHR per Branch (Gain/Loss?)

How large: $k$? Mostly $K=2$, $m=12$, how large $m$?

$2^p$ $11 \ldots 1$ $\text{BHT}$

$2^m$ $00 \ldots 00$ $11 \ldots 11$ $\text{PHT}$

$(\text{PC} \gg 2) \& (2^p - 1)$
Set of Branches – One Register

m bit

$2^p$

11 ..... 1 0

BHT

$(PC \% 2^p)$
What if One Branch -> One History -> One PHT?

BHT

\[(PC >> 2) \& (2^p - 1)\]

PHT

\[00 \ldots 00\]
\[00 \ldots 01\]
\[00 \ldots 10\]
\[11 \ldots 11\]
GShare

For a given history and for a given branch (PC) counters are trained.
Y & P Classification [MICRO 91]

- **GAg**: Global History Register, Global History Table
- **PAg**: Per-Address History Register, Global History Table
- **PAp**: Per-Address History Register, Per-Address History Table