# Lecture 20: Speculative execution

CMPE 120: Computer Organization & Architecture

Olivia Weng

# Logistics

- Reminder: check-in with professor is 1% of your final grade
  - Sign up link

- Grade remediation plan

# Grade remediation plan

- HW1 Quiz Remediation: meet with professor for 15 minutes
  - Bring a newly hand-written solution. Explain what you got wrong on your quiz and your new solution.
  - Replaces HW1 quiz grade & drops lowest HW Quiz grade
  - Schedule a time on my calendar in the next ~3 weeks (10/30 - 11/20)
    - Link: https://calendar.app.google/d3D4hm3JbmpXcrVt5
    - 100 appointment slots available for 67 students
    - If none of these times work, email me to schedule a time during this window.

- Final exam grade can replace the midterm grade if higher
  - → Midterm grade replacement = max(final exam grade, midterm exam grade)

# Grade remediation plan

- How HW1 Quiz remediation will go
  - Bring paper w/your handwritten solution + any handwritten notes needed for your explanation
  - Explain your mistakes to me
  - Explain your solution to me line-by-line w/sample input parameter
  - Professor will give feedback

- HW1 Quiz remediation score based on the quality of your explanation
  - Pass: Replace HW1 quiz grade & drop lowest HW Quiz grade
  - Partial pass: Only drop lowest HW quiz grade
  - No pass: Neither

# Example remediation

```
unsigned int get_lowest_4bits(unsigned int x) {
    return x & 0xF;
}
```

# Example remediation score

Old score:

HW1 Quiz: 0 / 2

HW2 Quiz: 1 / 2

HW3 Quiz: 0 / 2

Midterm: 20 / 70

Projected final grade: F

New score w/Remediation Pass:

HW1 Quiz: 0 / 2 → replaced with 2/2

HW2 Quiz: 1 / 2

~~HW3 Quiz: 0 / 2~~ Dropped!

Midterm: ~~20 / 70~~ → replaced with 85%

HW4 Quiz: 1 / 2

HW5 Quiz: 2 / 2

Final: 85%

Final grade: B

# Example remediation score

Old score:

HW1 Quiz: 0 / 2

HW2 Quiz: 1 / 2

HW3 Quiz: 0 / 2

Midterm: 20 / 70

Projected final grade: F

New score w/Remediation Partial Pass:

HW1 Quiz: 0 / 2

HW2 Quiz: 1 / 2

~~HW3 Quiz: 0 / 2~~ Dropped!

Midterm: ~~20 / 70~~ → replaced with 77%

HW4 Quiz: 1 / 2

HW5 Quiz: 2 / 2

Final: 77%

Final grade: C-

# How to make the most of this plan and pass this class

- Sign up for the HW1 quiz remediation meeting sooner rather than later

- Practice the questions you got wrong on midterm. Go over the ones you're still confused on w/me.
  - If you're not free during my office hours, email me to schedule an appointment.

- Do the practice midterm & final exam. Do the problems you struggle with multiple times without aids.

# How to make the most of this plan and pass this class

- Work through HW3 completely on your own without AI (turn it all off!)
  - → The HW4 and HW5 quizzes will mostly focus on the concepts from HW3

- Attend TA homework discussion sessions
  - Fridays: 1 - 2pm & 6 - 7pm on Zoom

# Speculative execution

- The problem is that we must wait for branches to resolve
  - Known as a control hazard

- But program behavior is predictable!

# Speculative execution

- The problem is that we must wait for branches to resolve
  - Known as a control hazard

- But program behavior is predictable!
  - Loops
  - If statements checking for unlikely error conditions
  - Unconditional branches, i.e., jumps

# Speculative execution

- Idea: Processor makes educated guess if branch is taken or not
  - Known as speculative execution or branch prediction
  - Guess in Fetch stage--no more hazard!

- Two key questions:
  - How to guess?
  - What to do when guess is wrong?

# What if I guess wrong?

- Branch condition not resolved until the Execute stage
  - These 2 instructions must be flushed if branch is taken, i.e., flush the pipeline

| | Cycle | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| `beq s1, s2, L1` | F | D | E | M | W | | | |
| `sub s8, t1, s3` | | F | D | E | M | W | | |
| `or s9, t6, s5` | | | F | D | E | M | W | |
| … | | | | | | | | |
| `L1: add s7, s3, s4` | | | | F | D | E | M | W |

# What if I guess wrong?

- Branch condition not resolved until the Execute stage

- Branch misprediction penalty: # of instructions that must be flushed

What is the branch misprediction penalty?

| | Cycle | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| `beq s1, s2, L1` | F | D | E | M | W | | | |
| `sub s8, t1, s3` | | F | D | E | M | W | | |
| `or s9, t6, s5` | | | F | D | E | M | W | |
| … | | | | | | | | |
| `L1: add s7, s3, s4` | | | | F | D | E | M | W |

A. 1 B. 2 C. 3 D. 4

# How to predict branches?

- Static branch prediction
  - Set policy that does not learn
  - Ex: Always predict taken

- Dynamic branch prediction
  - Can we learn from the past and use it to predict the future?

# Dynamic branch prediction

- 1-bit branch predictor: guess branch will take same direction as last time
  - Also known as last time predictor

# Dynamic branch prediction

- 1-bit branch predictor: guess branch will take same direction as last time
  - Also known as last time predictor
  - Ex: Last time branch was taken → Predict taken
  - Ex: Last time branch was not taken → Predict not taken

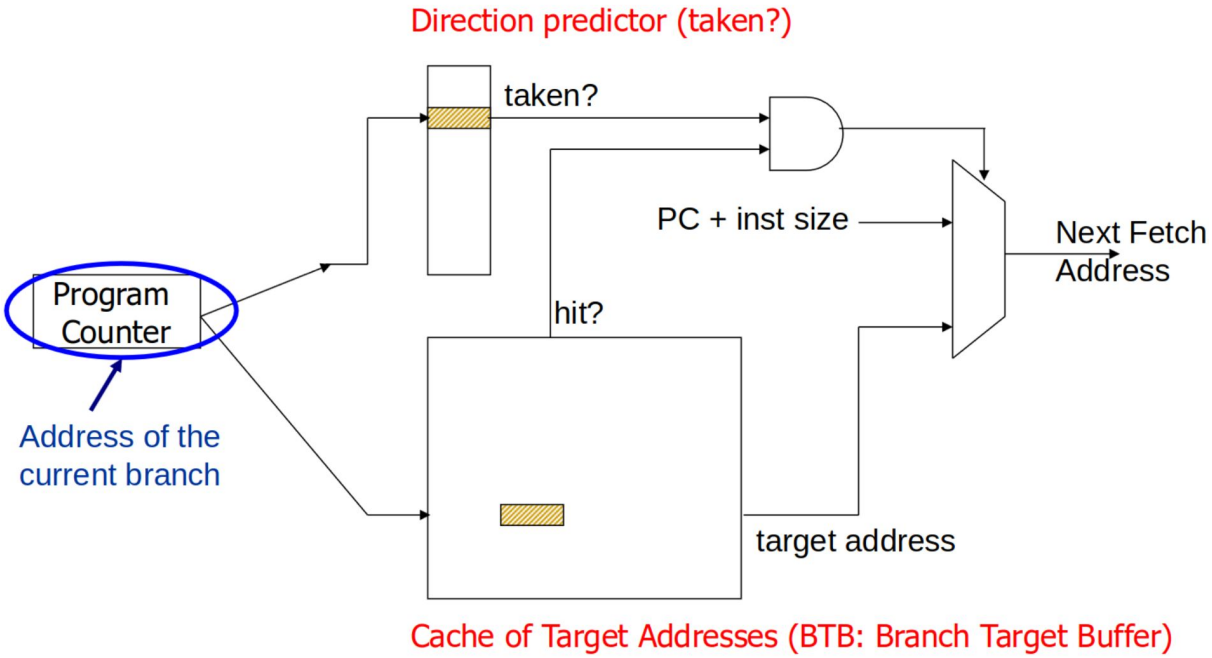
# What is needed to implement 1-bit branch predictor?

- We know the direction (taken vs not taken).

- We also need to know the branch target (i.e, where to jump)
  - So that we can update the PC correctly

# How to know where to jump to?

- Idea: store target address from last time and access it with the PC
  - Called the Branch Target Buffer (BTB)



# Improving the 1-bit predictor

- Problem: 1-bit predictor changes its prediction from T→NT or NT→T too quickly
  - Even when a branch may be mostly taken or mostly not taken

- Idea: Change prediction more slowly with 2-bit predictor
  - This way do not change prediction on a single different outcome

# 2-bit predictor

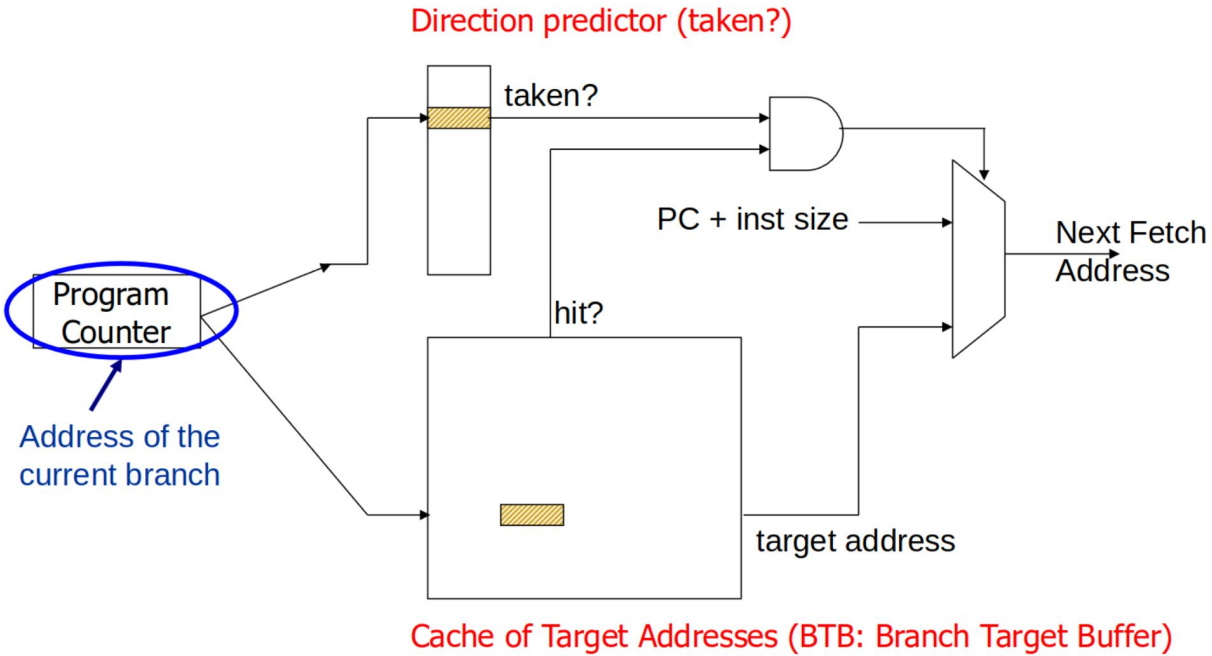
- Each branch associated with a 2-bit predictor
  - 00 → predict not taken (strongly not taken)
  - 01 → predict not taken (weakly not taken)
  - 10 → predict taken (weakly taken)
  - 11 → predict taken (strongly taken)

# 2-bit predictor

- Each branch associated with a 2-bit counter
  - Better prediction accuracy (~85-90%) than 1-bit counter for many programs

# 2-bit predictor

- Each branch associated with a 2-bit counter
  - Better prediction accuracy (~85-90%) than 1-bit counter for many programs



# Can we do better than 2-bit predictor?

- Note: a branch's outcome can be correlated with other branches' outcomes
  - Known as global branch correlation

```
if (condition1)

...

if (condition1 && condition2)
```

- If first branch not taken, second branch also not taken

# Can we do better than 2-bit predictor?

- Note: a branch's outcome can be correlated with other branches' outcomes
  - Known as global branch correlation

```
if (x < 1)

...

if (x > 1)

...
```

- If first branch taken, second definitely not taken

# Can we do better than 2-bit predictor?

- More examples?

# Can we do better than 2-bit predictor?

- More examples? If first branch taken, second definitely not taken

```
int a = 0;
if (condition1) {
    a = 2;
}
if (a == 0)
...
```

# Can we do better than 2-bit predictor?

- More examples? If first & second branch taken, third taken

```
if (condition1)

…

if (condition2)

…

if (condition1 && condition2)
```

# Can we do better than 2-bit predictor?

- Idea: Predict based on the global branch history
  - Take a broader view of the past to predict the future
  - Keep track of the global history in Global History Register (GHR)

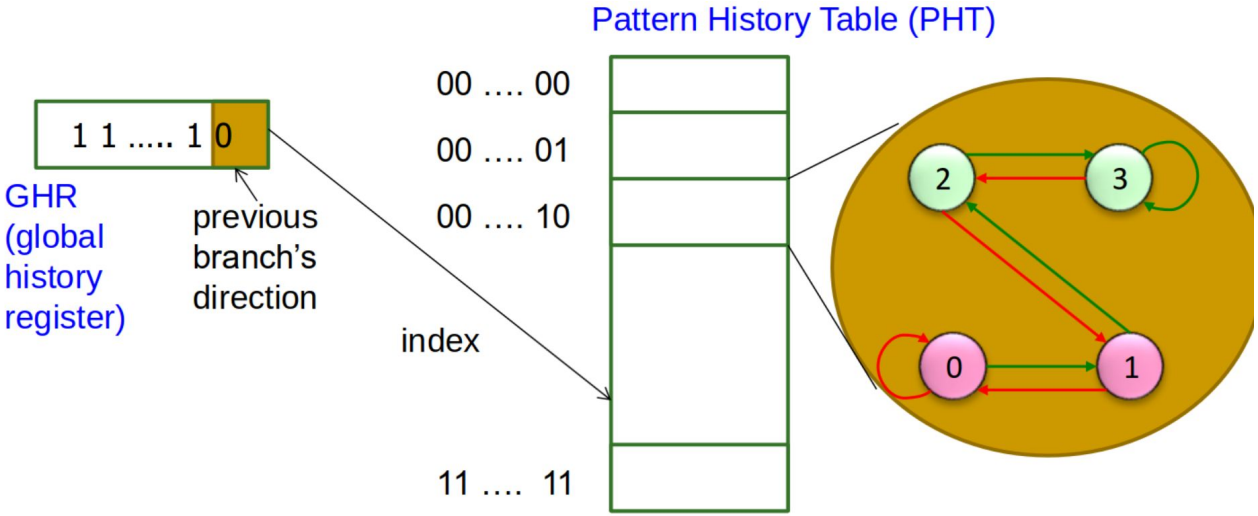
# Can we do better than 2-bit predictor?

- Idea: Predict based on the global branch history
  - Take a broader view of the past to predict the future
  - Keep track of the global history in Global History Register (GHR)

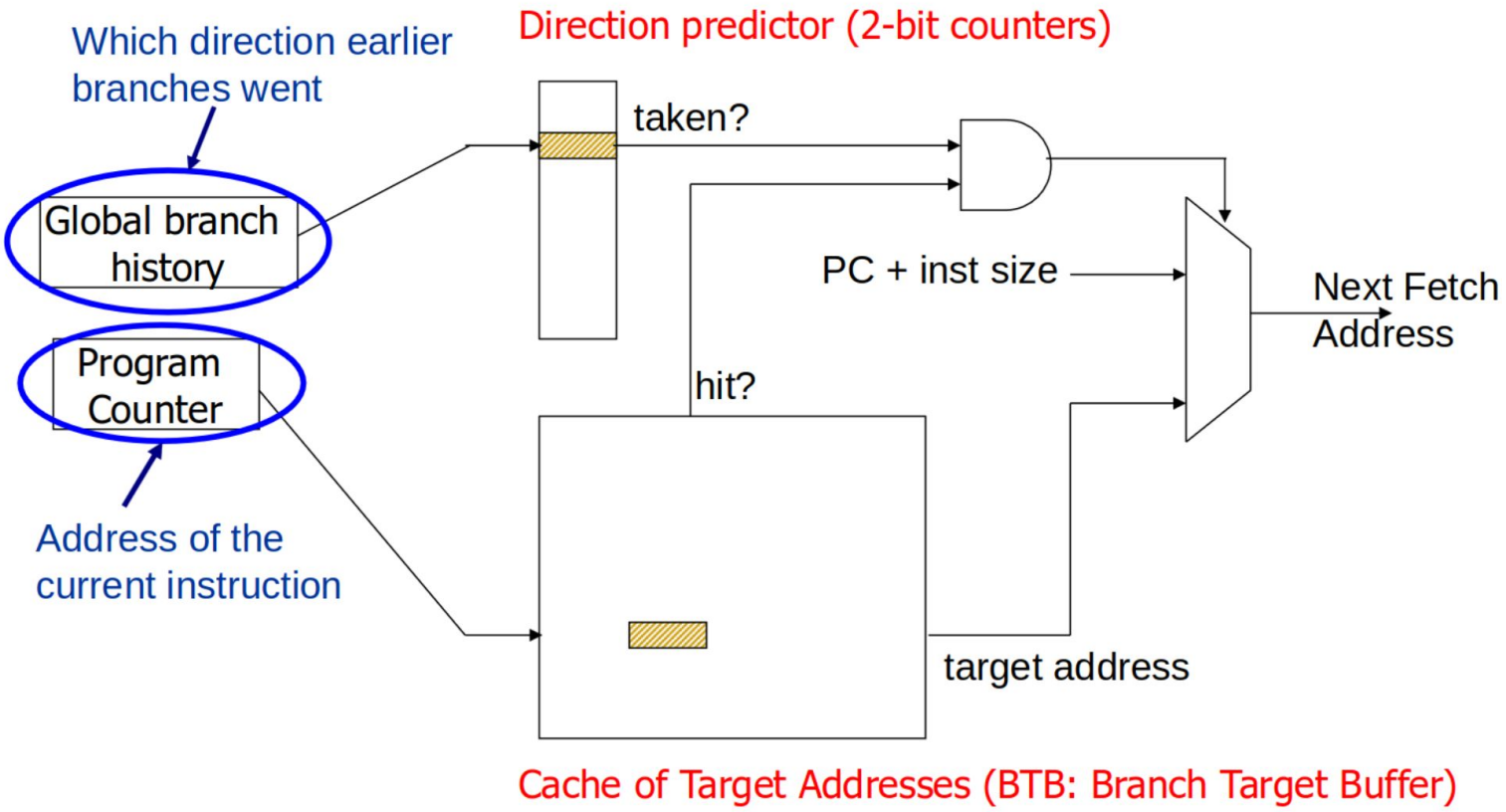
- Use GHR to index into a table that recorded the branch outcome for each GHR value recently
  - Table known as Pattern History Table (PHT)
  - Outcome recorded as 2-bit counter

# Two-level global branch prediction

- Level 1: Global branch history register (n bits)
  - Direction of the last n branches
- Level 2: Pattern history table of 2-bit counters for each history entry
  - Direction branch took last time same history was seen

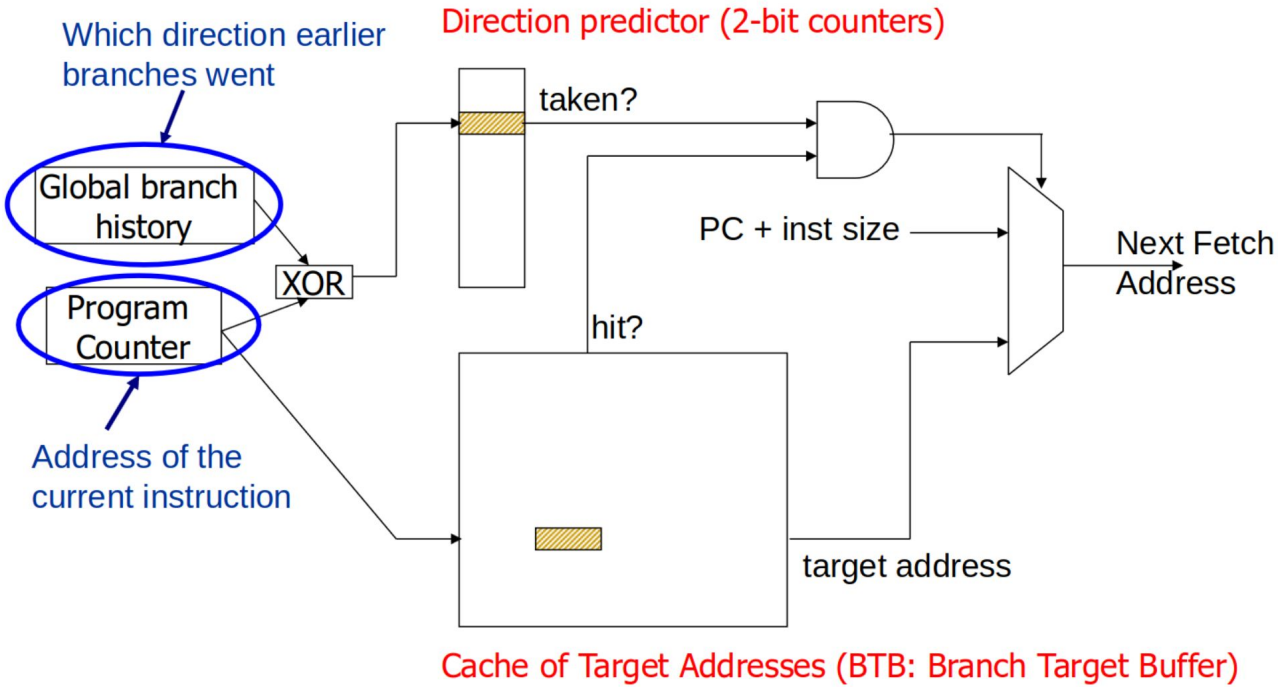


# Two-level global history brnach predictor



# Two-level gshare branch predictor

- gshare: global shared pattern history table greatly reduces mispredictions




Paper: https://american.cs.ucdavis.edu/academic/readings/papers/mcfarling.pdf