CMPE 200
Computer Architecture & Design

Lecture 3.
Processor Microarchitecture and Design (5)

Haonan Wang

SAN JOSÉ STATE
UNIVERSITY

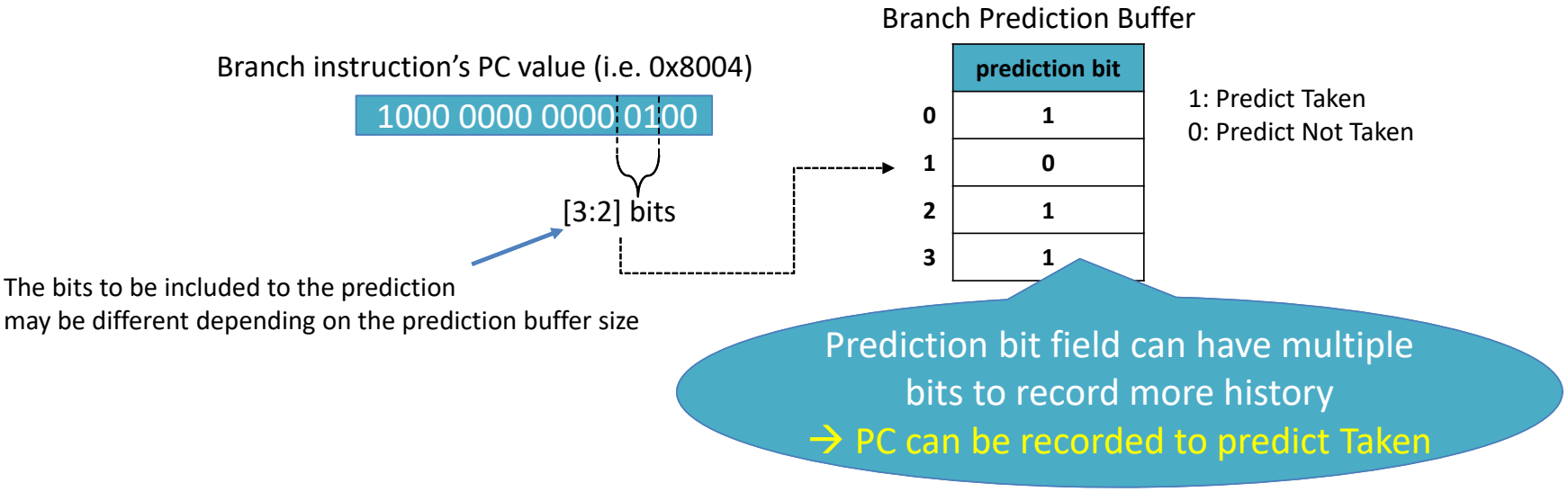
Dynamic Branch Prediction

- **Monitor the branch patterns for better prediction**
  - i.e. loops usually have a few iterations, so the branch in the loop is likely to be not taken at least for a few times

- **Branch Prediction Buffer (branch history buffer) is used to maintain each branch's outcome history**
  - Small storage space indexed with certain bits of branch instruction's PC value
  - Each entry maintains the previous outcome of each branch
  - Once the branch outcome is available, the corresponding entry is updated

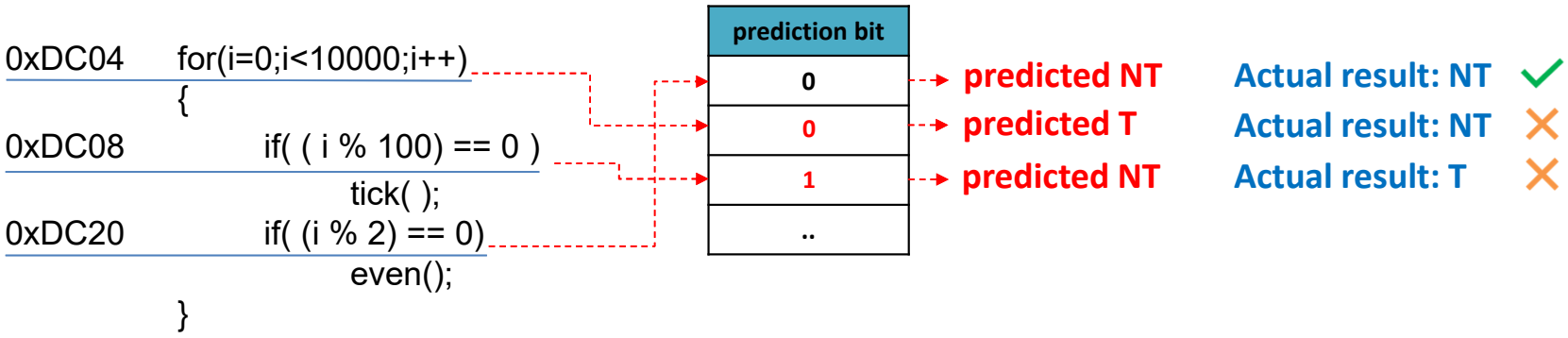
Branch Prediction Buffer

- **Aliasing issue**
  - Different branch instructions may map to the same branch prediction buffer entry if their PC values have the same LSBs, affecting each others' predictions
    - E.g., 0x8004 and 0x8104 will access the same entry if [3:2] bits are used for indexing
    - Inevitable but can be alleviated with larger buffer



1-bit Predictor

- **Each entry of branch prediction buffer is 1-bit (1: Taken, 0: Not Taken)**
  - The entry value is the latest outcome of the branch
  - Next outcome of the branch is predicted based on current value
  - Example: Assume that the actual outcomes of the branches at **0xDC04**, **0xDC08**, and **0xDC20** are **untaken**, **taken**, and **untaken**, respectively



Is 1 bit Enough?

```
0xDC04    for(i=0;i<10000;i++)
          {
0xDC08          if( ( i % 100) == 0 )
                    tick( );
0xDC20          if( (i % 2) == 0)
                    even();
          }
```

Not taken and continue to the loop body

Taken to exit the loop

NT

TN

DC04: NNNNNNN ... NNNNNNNNNNNTNNNNNNNNN ...

10,000 iterations

10,000 iterations ...

Mis-predictions for every first and last iterations → **99.998% Correct Prediction**

Mis-predictions: 2 / 10,000

DC08: TTTTT ... TNTTTTT ... TNTTTTT ...

98.0% Correct Prediction

Mis-predictions: 2 / 100

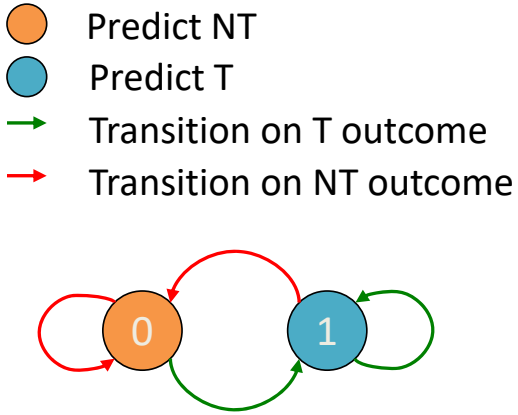
DC20: TNTNTNTNTNTNTNTNTNTNTNTNTNTNT ...

0.0% Correct Prediction

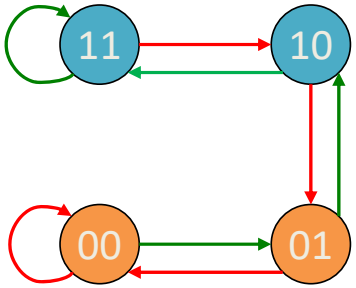
Mis-predictions: 2 / 2

Using 2-bit History

- **2-bit Saturating Counter in each branch prediction buffer entry**
  - Could have more than two bits but two bits cover most patterns (i.e. loops)

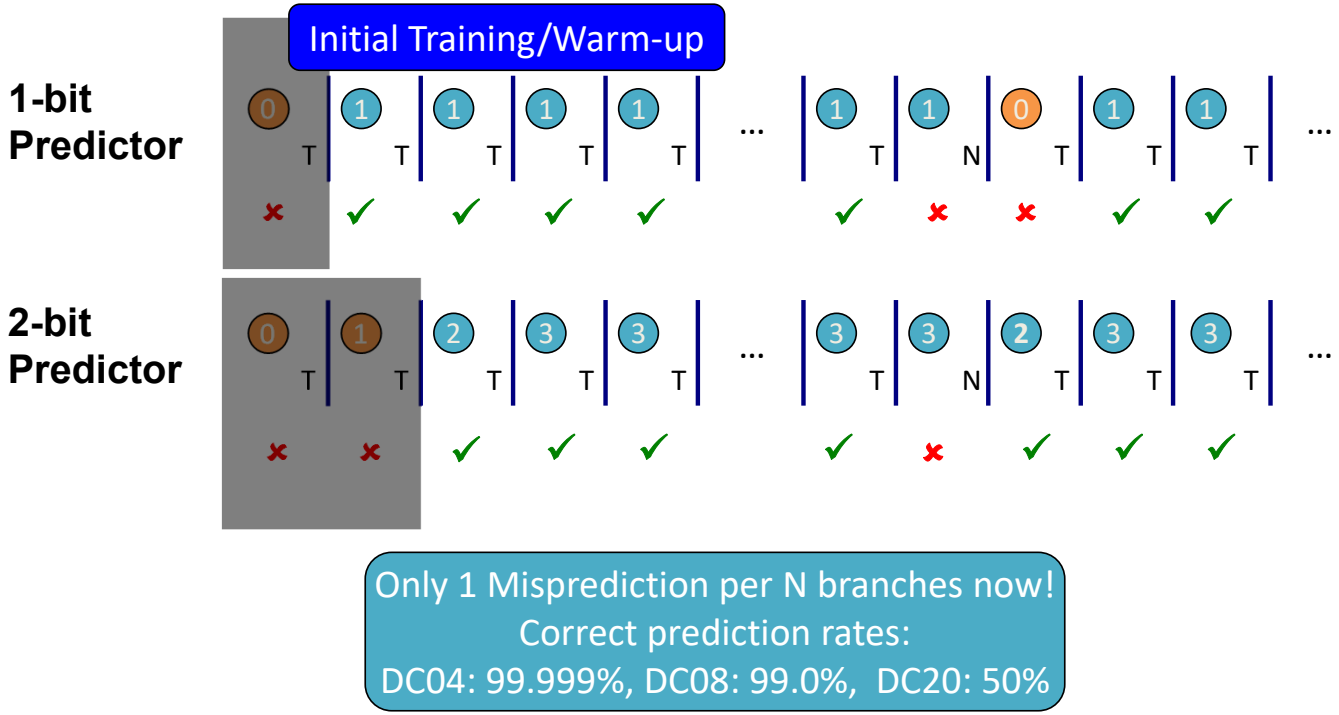


FSM for 1-bit Prediction



FSM for 2-bit Saturating Counter

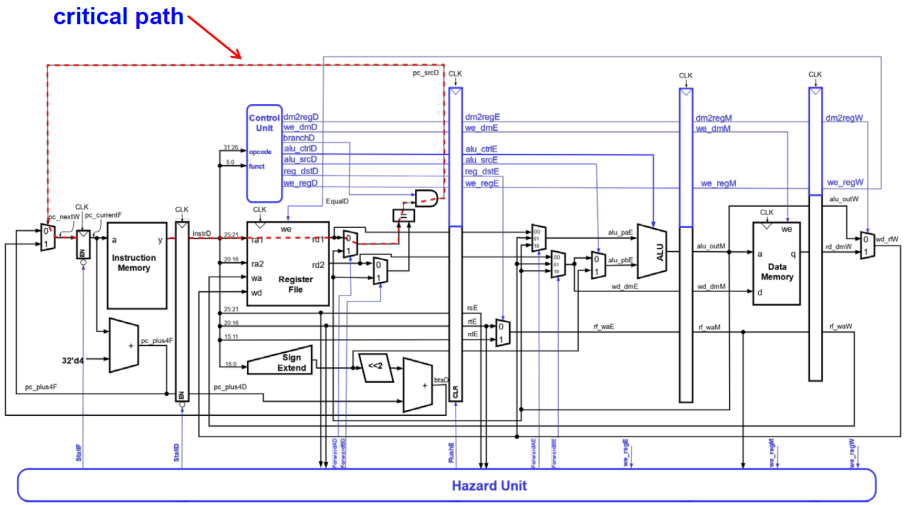
Example



Only 1 Misprediction per N branches now!
Correct prediction rates:
DC04: 99.999%, DC08: 99.0%, DC20: 50%

Pipelined CPU Performance Analysis

Function Unit	Parameter	Delay (ps)
Register clock-to-Q	T_{pcq}	30
Clock setup time	T_{setup}	20
MUX	T_{MUX}	25
Sign-extend	T_{s_ext}	25
ALU	T_{ALU}	200
Mem read	T_{mem}	250
AND	T_{AND}	15
EQ	T_{eq}	40
Register file read	T_{RFread}	150
Register file write	$T_{RFwrite}$	20



Critical path = max {

$T_{pcq} + T_{i\text{-}mem} + T_{setup}$,

2 ($T_{RFread} + T_{mux} + T_{eq} + T_{AND} + T_{mux} + T_{setup}$ **)**,

$T_{pcq} + T_{mux} + T_{mux} + T_{ALU} + T_{setup}$,

$T_{pcq} + T_{memread} + T_{setup}$,

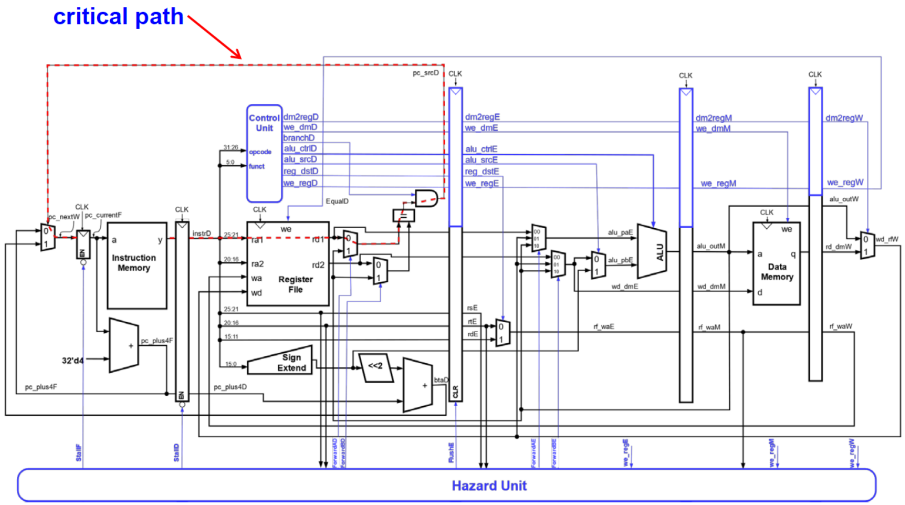
2 ($T_{pcq} + T_{mux} + T_{RFwrite}$ **)**

}

RF write and read need to be finished in the first and second half of each cycle, respectively

Pipelined CPU Performance Analysis

Function Unit	Parameter	Delay (ps)
Register clock-to-Q	T_{pcq}	30
Clock setup time	T_{setup}	20
MUX	T_{MUX}	25
Sign-extend	T_{s_ext}	25
ALU	T_{ALU}	200
Mem read	T_{mem}	250
AND	T_{AND}	15
EQ	T_{eq}	40
Register file read	T_{RFread}	150
Register file write	$T_{RFwrite}$	20



Critical path = **2 (** $T_{RFread} + T_{mux} + T_{eq} + T_{AND} + T_{mux} + T_{setup}$ **)**

= 2 (150 + 25 + 40 + 15 + 25 + 20) = **550 ps**

Pipelined CPU Performance Analysis

- **SPECINT2000 benchmark:**
  - 25% loads, 10% stores, 11% branches, 2% jumps, 52% R-type
- **Assume:**
  - 40% of loads used by next instruction
  - 25% of branches mispredicted
  - All jumps flush next instruction
- **Average CPI calculation:**
  - Load/Branch CPI = 1 when no stalling, 2 when stalling
  - $CPI_{lw} = 1(0.6) + 2(0.4) = 1.4$
  - $CPI_{beq} = 1(0.75) + 2(0.25) = 1.25$
  - **Overall average CPI = (0.25 x 1.4) + (0.1 x 1) + (0.11 x 1.25) + (0.02 x 2) + (0.52 x 1) = 1.15**

Pipelined CPU Performance Analysis

- **For a program with 100 billion instructions executing on the pipelined MIPS processor under discussion, execution time is**

CPU time = # instructions x CPI x cycle time

= (100 x 10^9) x 1.15 x (550 x 10^{-12} sec)

= 63 seconds

- **Recall: when we ran the same code on a single-cycle processor (Lecture 4 – (3)), CPU time was 92.5 seconds.**

Speedup of pipelined architecture = 92.5 / 63 = 1.47x

Question

Suppose we are writing code for a machine with unknown branch predictor. We found that certain loops containing branch instructions have long execution times. What can we do?

- **Swap the branches**

SAN JOSÉ STATE UNIVERSITY *powering* SILICON VALLEY