Topic V2A

Control Hazards

Readings: Section(4.9)

Control Hazards

```
concatenate:
    lb      t0, 0(a0)          #
    beq     t0, zero, append   #
nextchar:
    addi    a0, a0, 1          #
    lb      t0, 0(a0)          #
    bne     t0, zero, nextchar
append:
    # a0 contains the position
    lb      t1, 0(a1)          #
    sb      t1, 0(a0)          #
    addi    a0, a0, 1          #
    addi    a1, a1, 1          #
    bne     t1, zero, append   #
    jr      ra
```

Not taken

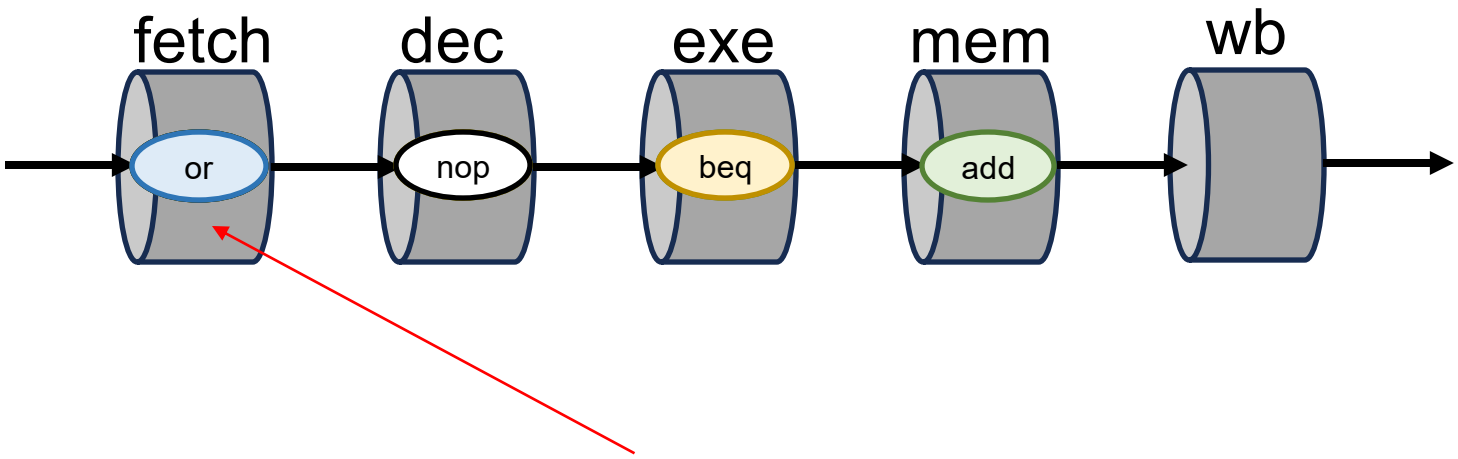
Taken

- Branch determines flow of control
  - Fetching next instruction depends on branch outcome
  - Pipeline can't always fetch correct instruction
    - Still working on ID stage of branch
- In RISC-V pipeline
  - Need to compare registers and compute target early in the pipeline
  - Add hardware to do it in ID stage

Control Hazard

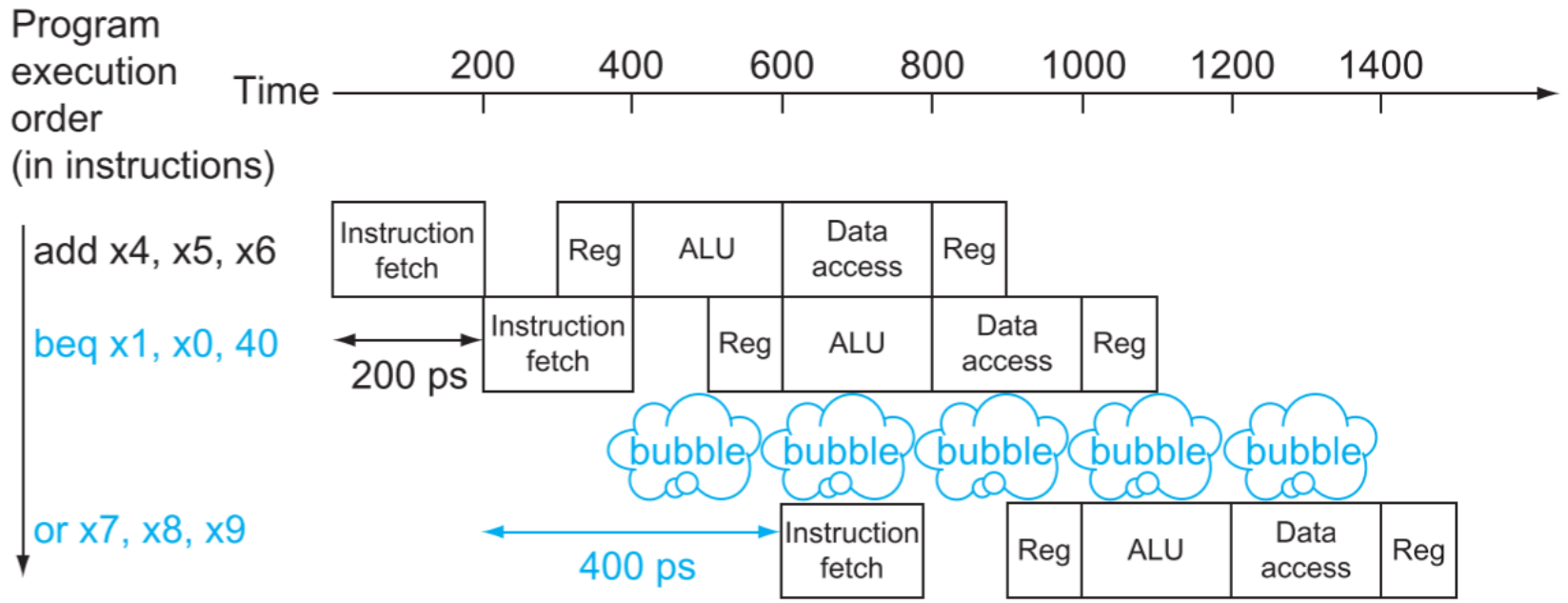
What instruction to fetch after a branch.

```
add    x4, x5, x6
beq    x1, x0, TARGET
or     x7, x8, x9
```



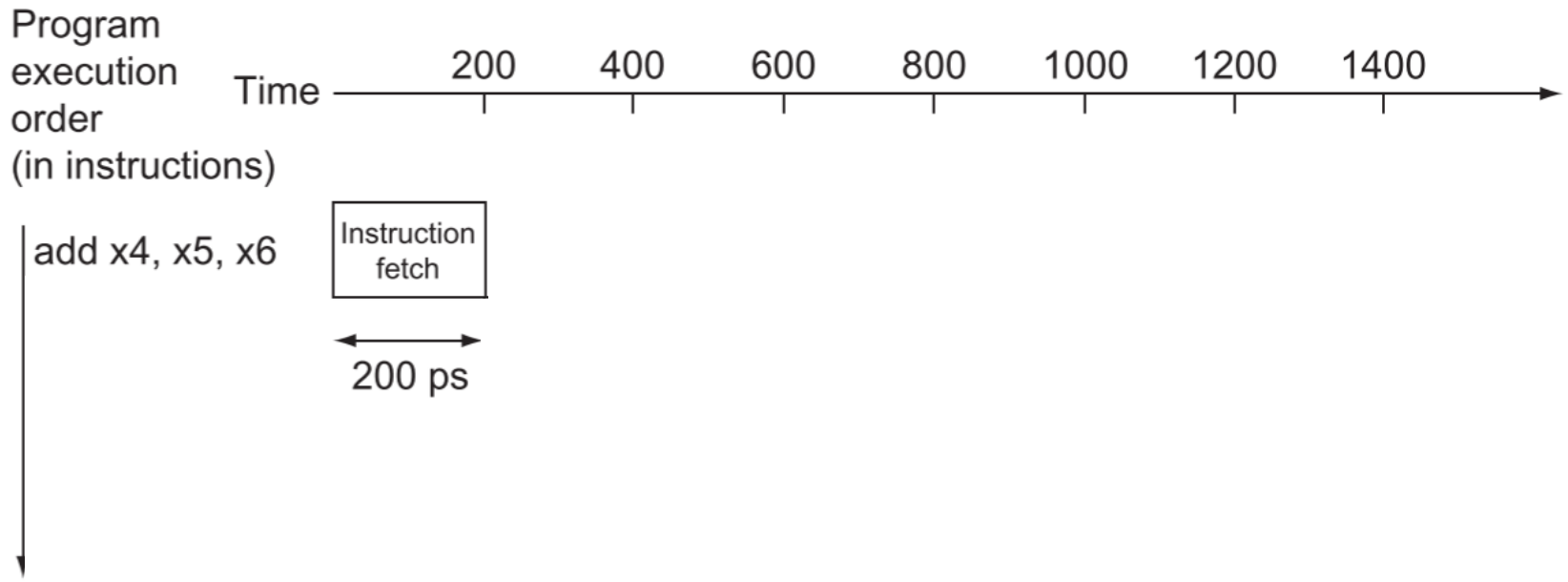
Stall on Branch

Wait until branch outcome is determined before fetching next instruction



Stall on Branch

Wait until branch outcome is determined before fetching next instruction

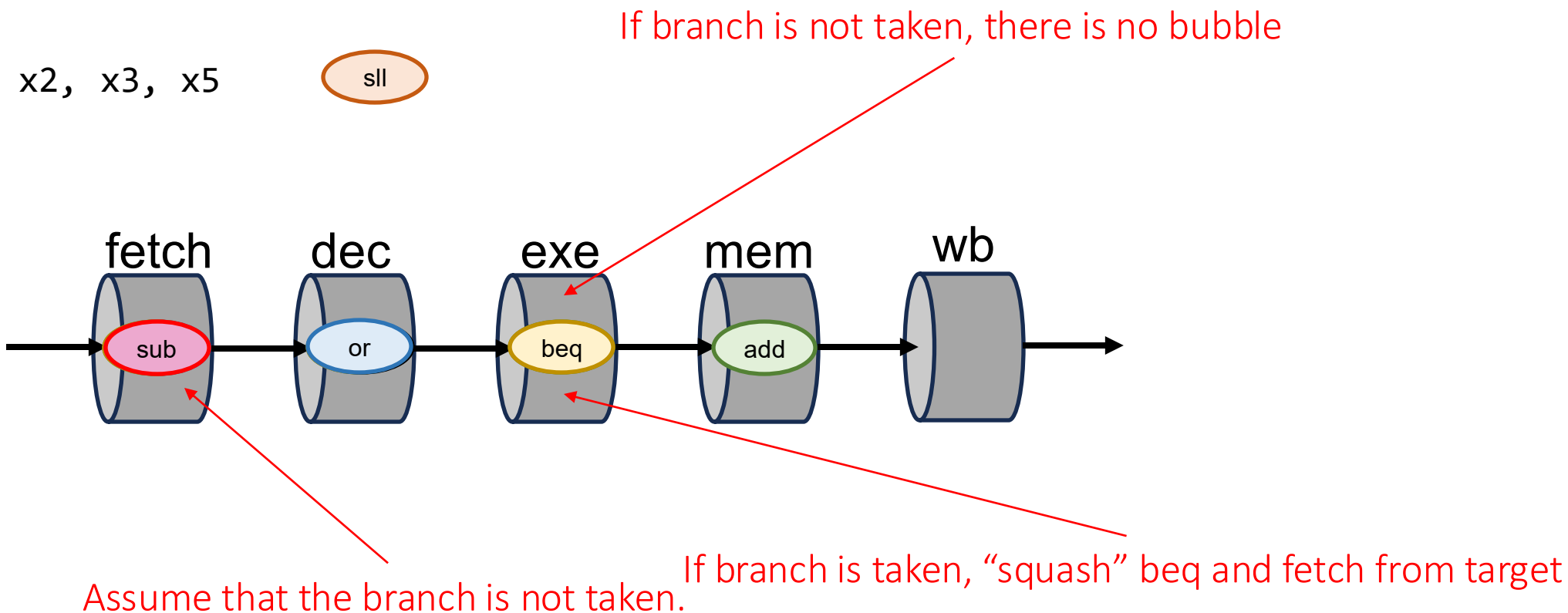


Branch Prediction

- Longer pipelines can’t readily determine branch outcome early
  - Stall penalty becomes unacceptable
- Predict outcome of branch
  - Only stall if prediction is wrong
- In RISC-V pipeline
  - Can predict branches not taken
  - Fetch instruction after branch, with no delay

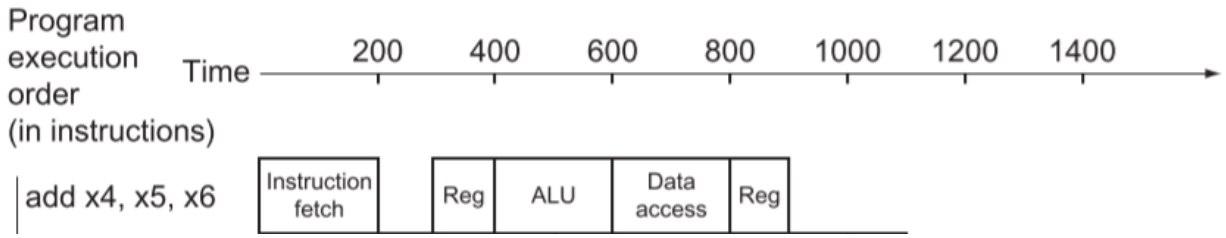
Predict Branches as Not Taken

```
        add   x4, x5, x6
        beq   x1, x0, SKIP
        or    x7, x8, x9
        sub   x1, x2, x3
        ….
SKIP:   sll x2, x3, x5
```



RISC-V with Predict Not Taken

Prediction correct



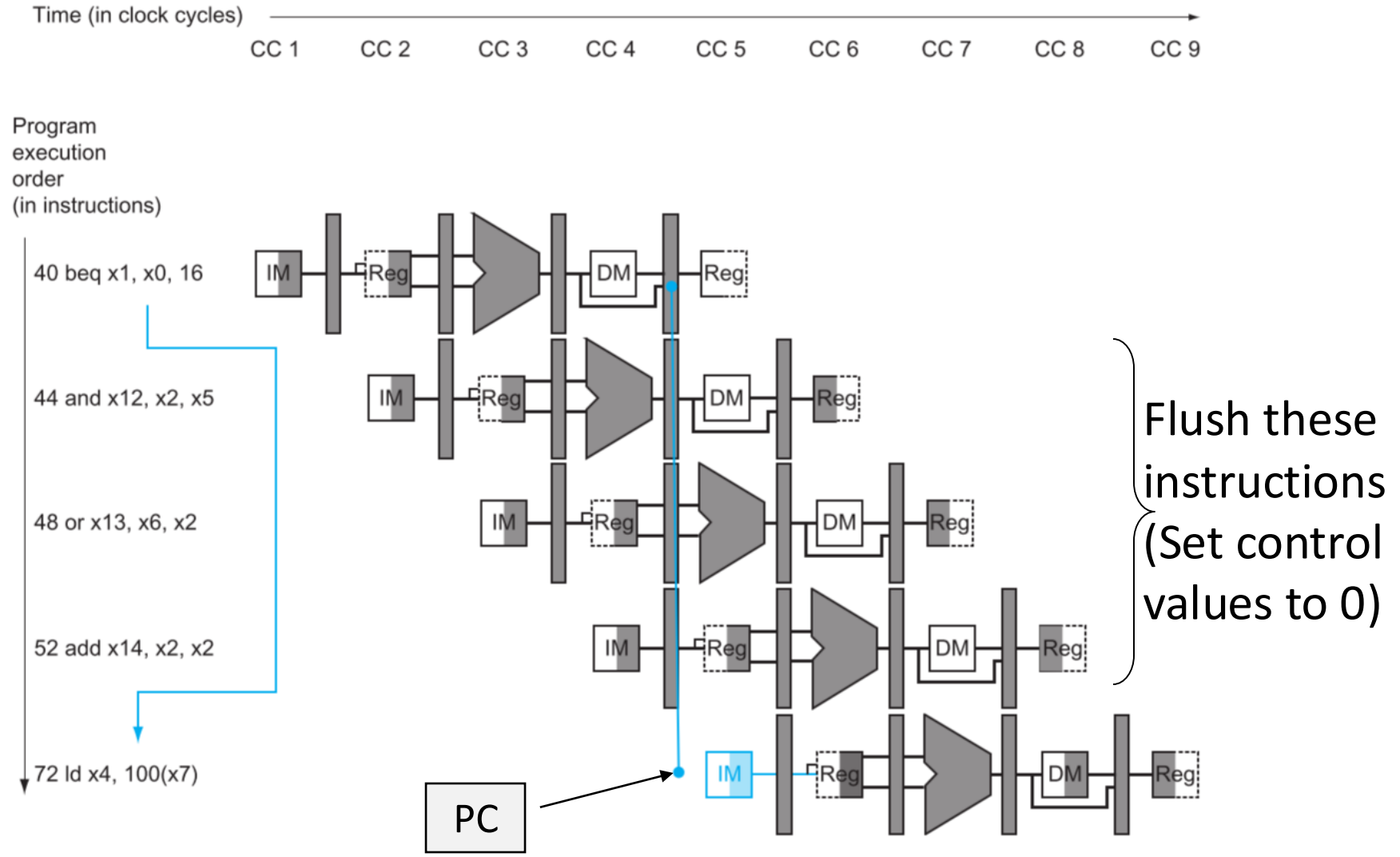
Prediction incorrect

More-Realistic Branch Prediction

- Static branch prediction
  - Based on typical branch behavior
  - Example: loop and if-statement branches
    - Predict backward branches taken
    - Predict forward branches not taken
- Dynamic branch prediction
  - Hardware measures actual branch behavior
    - e.g., record recent history of each branch
  - Assume future behavior will continue the trend
    - When wrong, stall while re-fetching, and update history

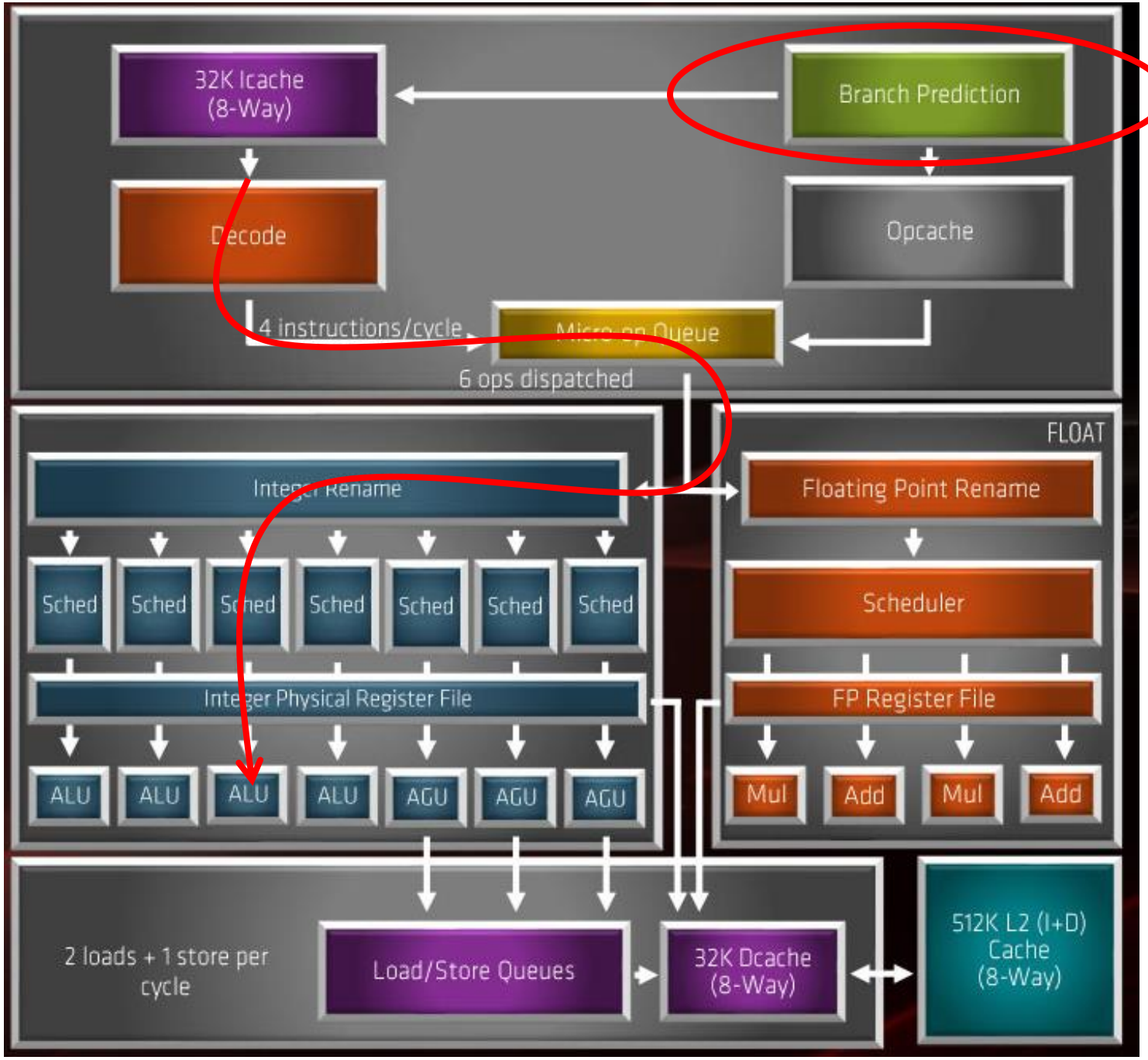
Branch Hazards

If branch outcome determined in MEM

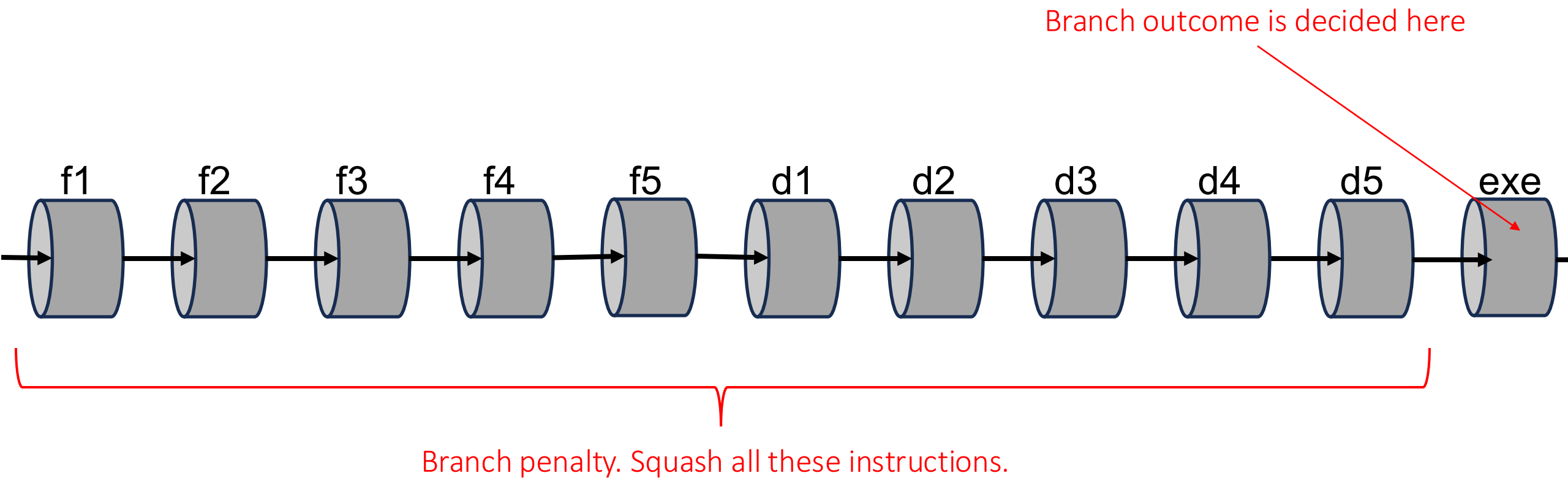



AMD Ryzen 9 3900X
AMD
RYZEN
100-000000023
BF 1922SUT
9HH0825R90088
DIFFUSED IN USA
DIFFUSED IN TAIWAN
MADE IN CHINA
© 2019 AMD
CPU1

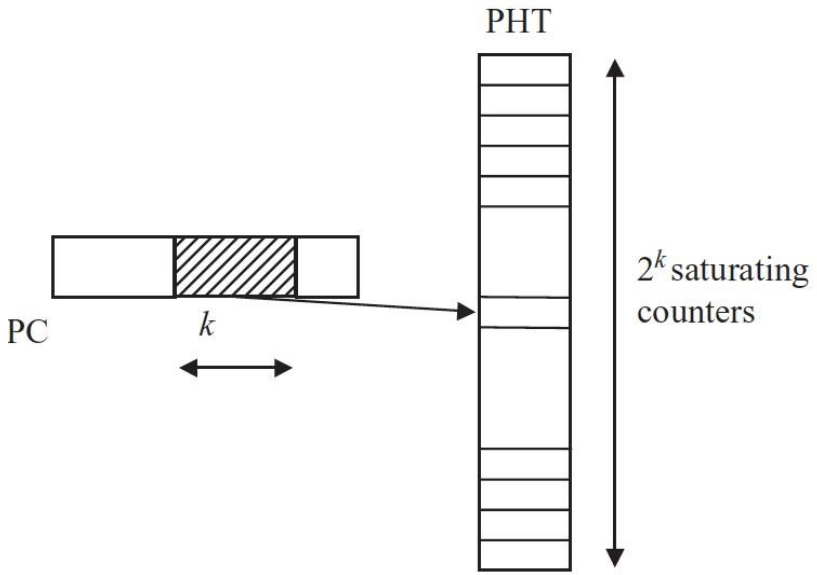

AMD
3D V-CACHE™
TECHNOLOGY
AMD
RYZEN
9
AMD
RYZEN
7000 SERIES PROCESSOR
9
AMD



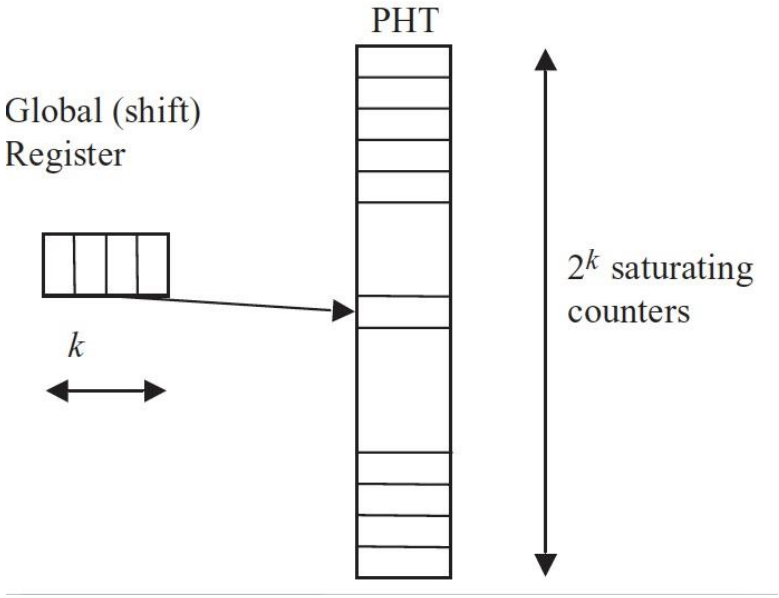
https://electronics.stackexchange.com/questions/732438/how-does-pipelined-cpu-access-both-code-and-data-memory-in-real-life


Branch outcome is decided here
f1
f2
f3
f4
f5
d1
d2
d3
d4
d5
exe
Branch penalty. Squash all these instructions.

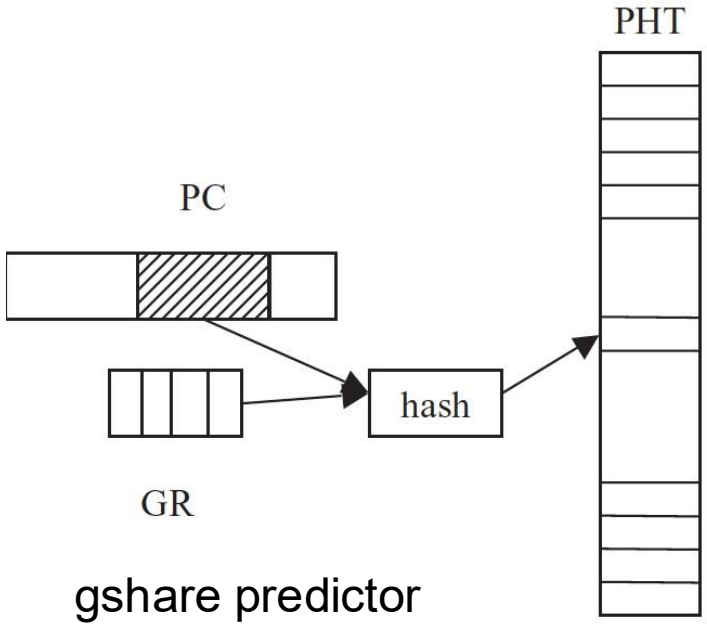
Dynamic Branch Prediction



bimodal predictor



global predictor



gshare predictor

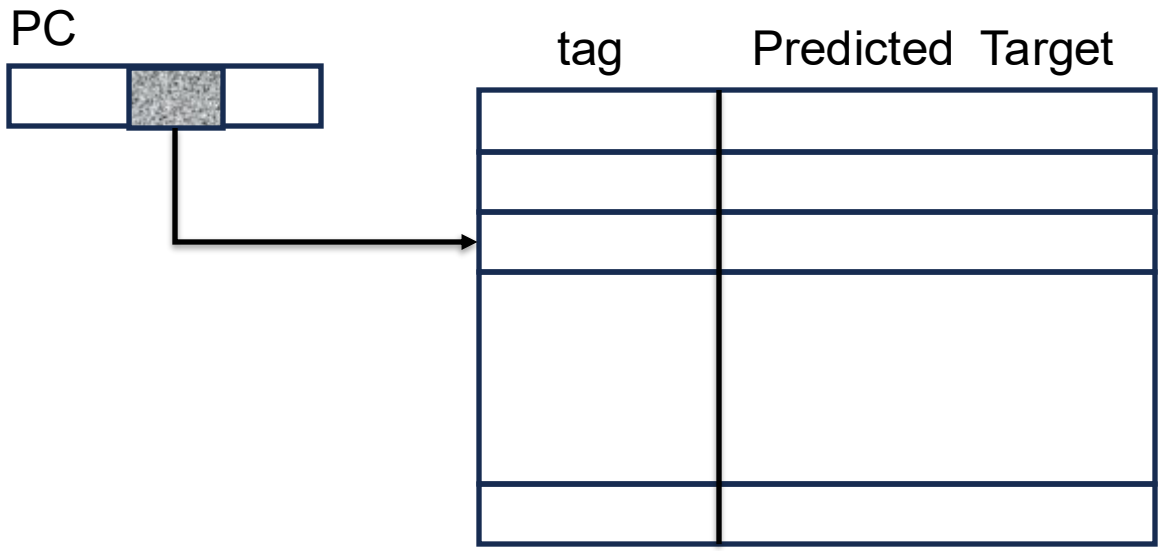
Dynamic Branch Prediction

In deeper and superscalar pipelines, branch penalty is more significant

Use dynamic prediction

- Branch prediction buffer (aka branch history table)
- Indexed by:
  - bits from the PC
  - recent branch outcomes (history)
  - combination of PC and history
- Stores outcome counter (taken/not taken)
- To execute a branch
  - Check table, expect the same outcome
  - Start fetching from fall-through or target
  - If wrong, flush pipeline and flip prediction

Predicting the Branch Target



Branch Target Buffer (BTB)

Calculating the Branch Target

- Even with predictor, still need to calculate the target address
  - 1-cycle penalty for a taken branch
- Branch target buffer
  - Cache of target addresses
  - Indexed by PC when instruction fetched
    - If hit and instruction is branch predicted taken, can fetch target immediately

Pipeline Summary

The BIG Picture

Pipelining improves performance by increasing instruction throughput

- Executes multiple instructions in parallel
- Does not reduce each instruction's latency

Subject to hazards

- Structure, data, control

Instruction set design affects complexity of pipeline implementation