COMPUTER ORGANIZATION (IS F242)

LECT 45: PIPELINING

Control Hazards or Branch Hazards

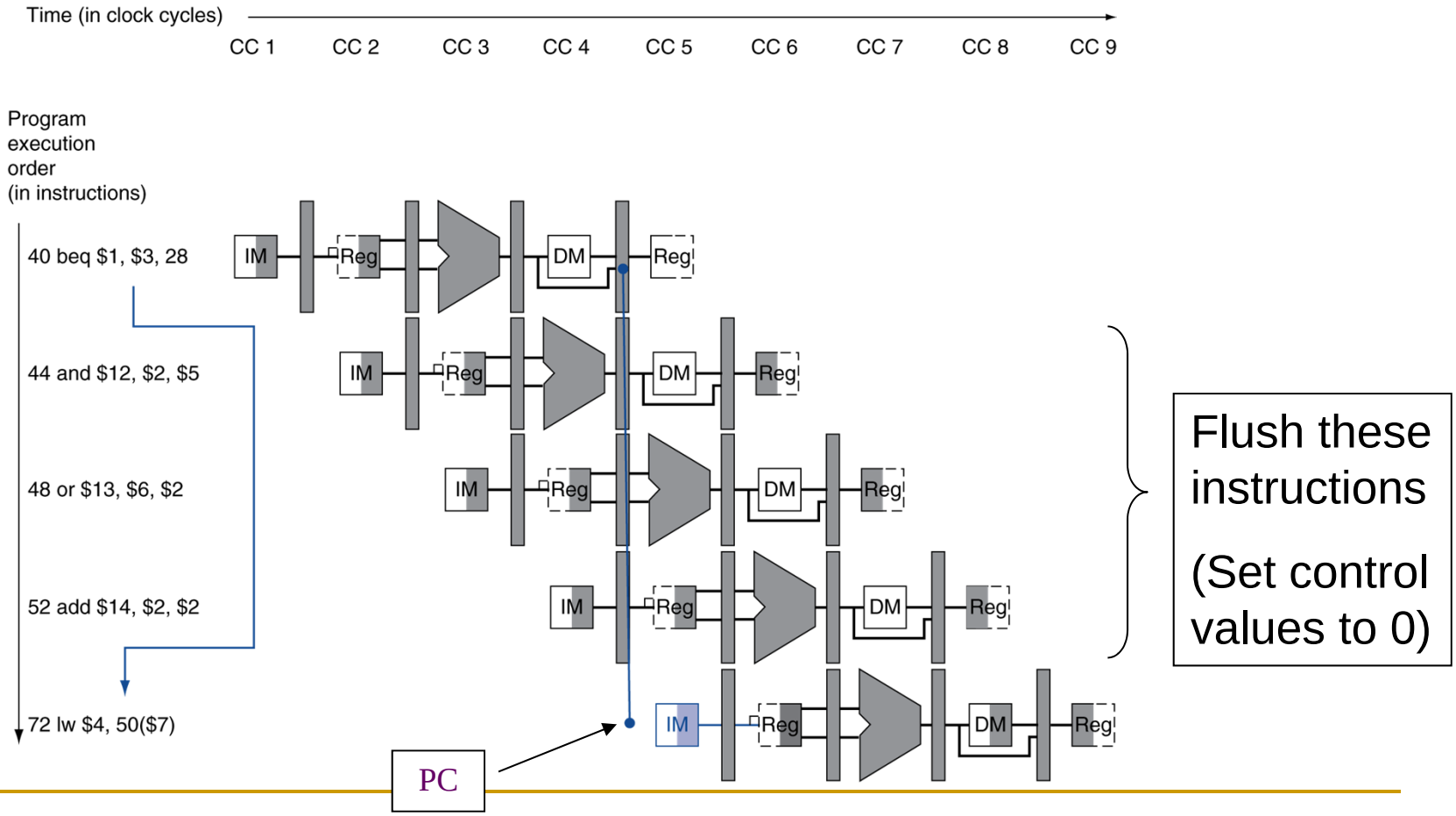
- Branch determines flow of control
  - Fetching next instruction depends on branch outcome
  - Pipeline can't always fetch correct instruction
    - Still working on ID stage of branch
- In MIPS pipeline
  - Need to compare registers and compute target early in the pipeline
  - Add hardware to do it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

Handling Conditional Branches in Pipeline

- Multiple Streams
  - Have two pipelines
  - Prefetch each branch into a separate pipeline
  - Use appropriate pipeline
  - Drawbacks
    - Leads to bus & register contention
    - Multiple branches lead to further pipelines being needed
    - Used in IBM 370/168
- Pre-fetch Branch Target
  - Target of branch is prefetched in addition to instructions following branch
  - Keep target until branch is executed
  - Used by IBM 360/91

Branch Hazards

- If branch outcome determined in MEM



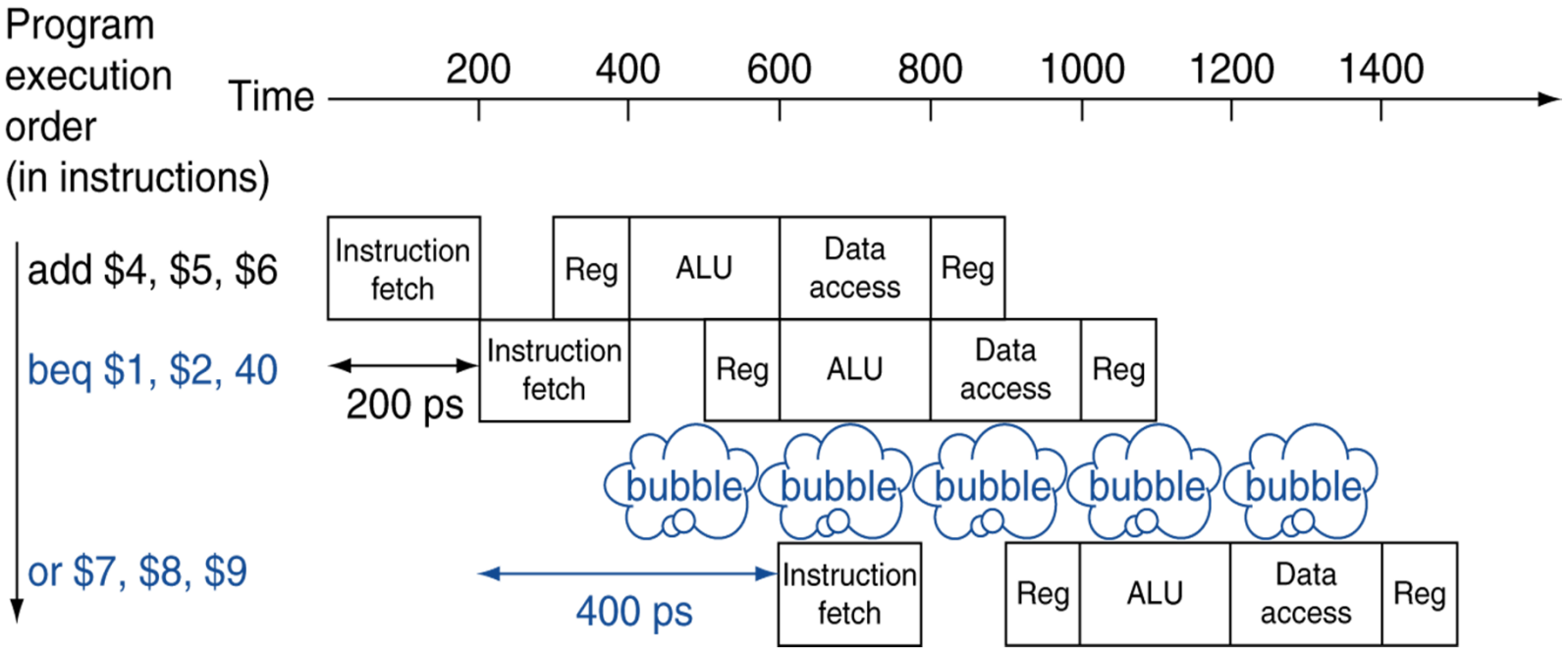
- In MIPS pipeline
  - Need to compare registers and compute target early in the pipeline
  - Add hardware to do it in ID stage

Reducing Branch Delay

- Move hardware to determine outcome to ID stage. Requires 2 actions to occur earlier
  - Computing branch target address
    - Move the branch target address calculation from EX to ID
    - PC and immediate field in IF/ID pipeline register is used for it
    - Disadvantage: branch target address calculation will be performed for all instructions (used only when needed)
  - Evaluating the branch decision
    - Branch equal
      - Compare 2 registers read during ID stage to see they are equal
      - XOR their respective bits and ORing all the results

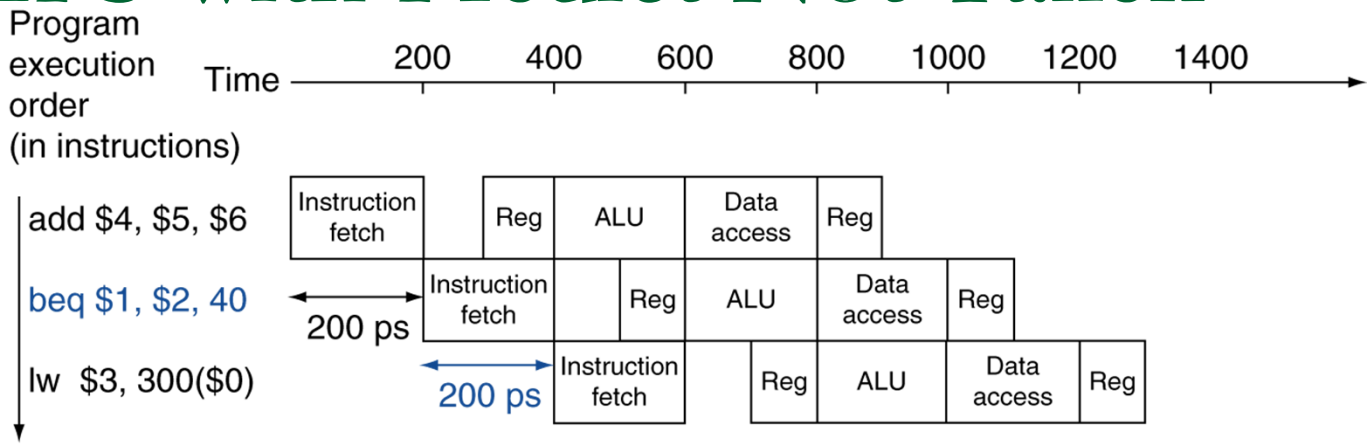
Stall on Branch

- Wait until branch outcome determined before fetching next instruction

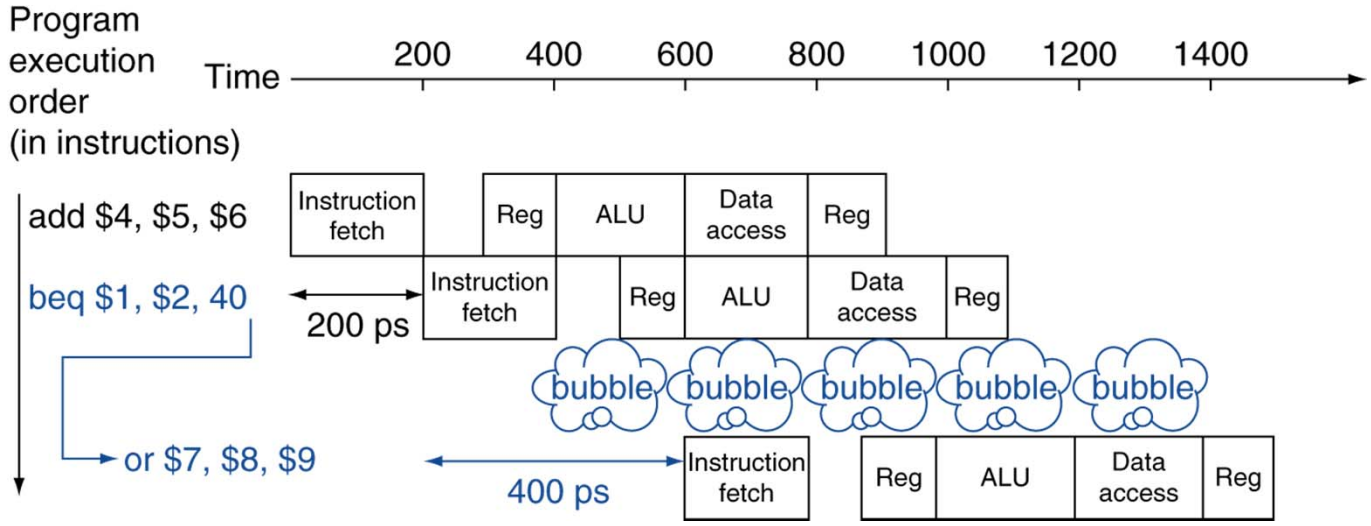


MIPS with Predict Not Taken

Prediction correct



Prediction incorrect

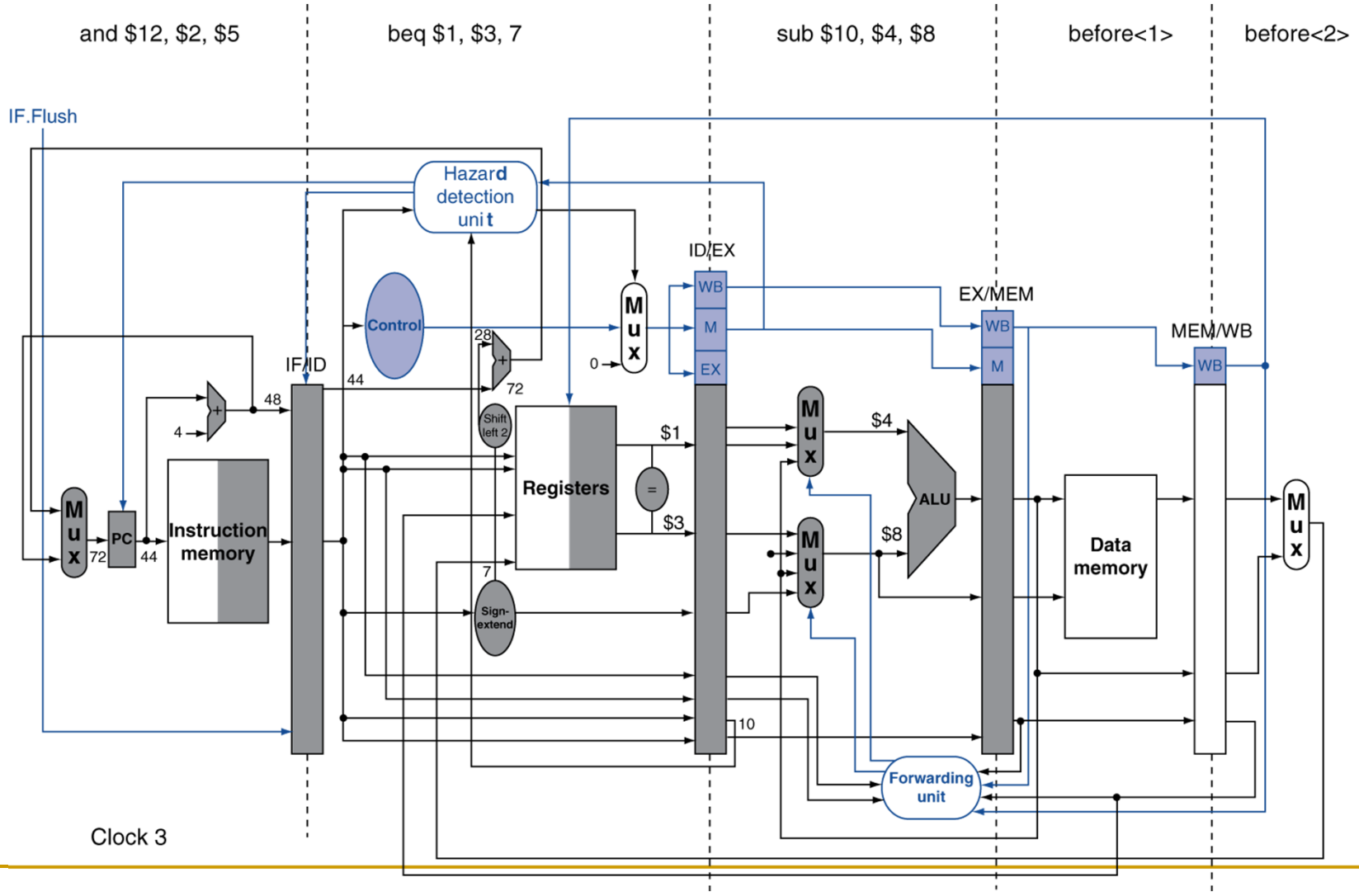


Pipelined Branch

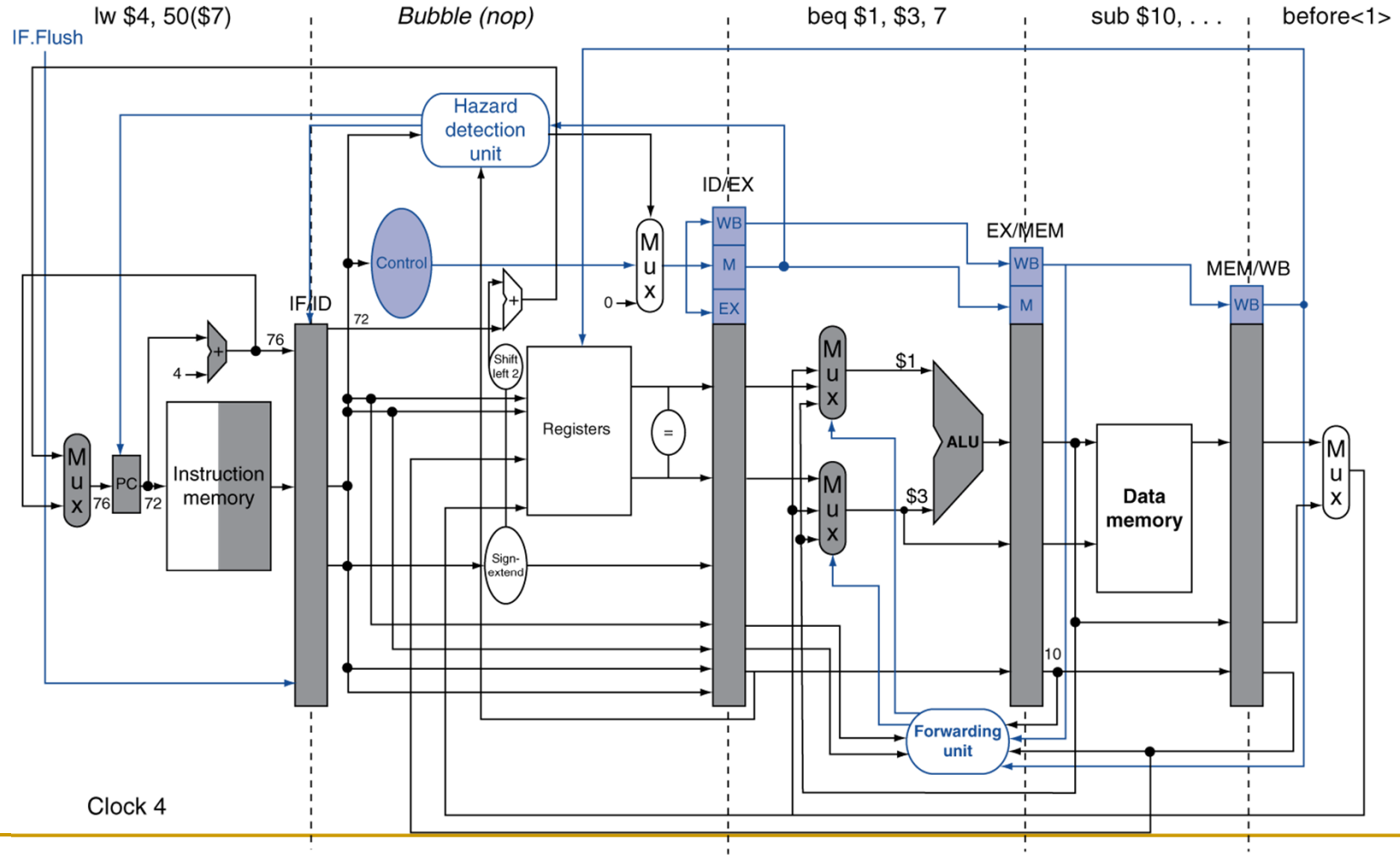
- Advantages of moving branch to ID stage
  - Reduces the penalty of branch to only one instruction if the branch is taken
- To flush instruction in control stage, a control line called IF.Flush is added
  - Zeroes the IF/ID pipeline register
- Example: branch taken

```
36:   sub   $10, $4, $8
40:   beq   $1,  $3, 7
44:   and   $12, $2, $5
48:   or    $13, $2, $6
52:   add   $14, $4, $2
56:   slt   $15, $6, $7
      ...
72:   lw    $4, 50($7)
```

Example: Branch Taken



Example: Branch Taken



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 MIPS pipeline
  - Can predict branches not taken
  - Fetch instruction after branch, with no delay

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

Handling Conditional Branches in Pipeline

- In deeper and superscalar pipelines, branch penalty is more significant
- Use dynamic prediction
  - Prediction of branches at runtime using runtime information

Dynamic Branch Prediction

- Predict always taken
  - Assume that jump will happen
  - Always fetch target instruction
- Predict by Opcode
  - Some instructions are more likely to result in a jump than others
  - Can get up to 75% success

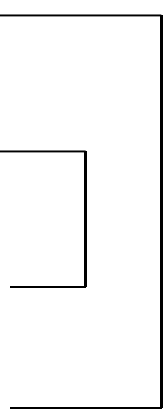
Dynamic Branch Prediction

- Use dynamic prediction
  - Prediction of branches at runtime using runtime information
  - Branch prediction buffer (aka branch history table)
  - Memory contains a bit that says whether branch was taken not.
    - Another branch with same lower order address bits will update this memory. So the prediction may not be the right one.
  - To execute a branch
    - Check table, expect the same outcome
    - Start fetching from fall-through or target
    - If wrong, flush pipeline and flip prediction

1-Bit Predictor: Shortcoming

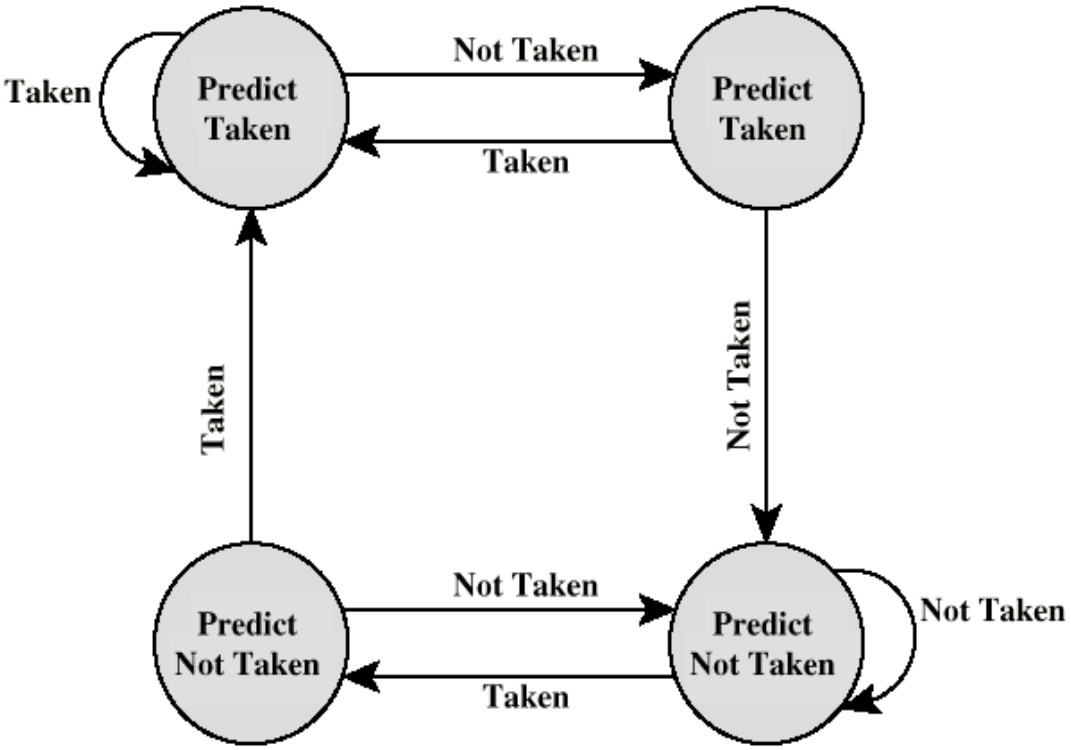
- Inner loop branches mispredicted twice!

```
outer: …
       …
inner: …
       …
       beq …, …, inner
       …
       beq …, …, outer
```



- Mispredict as taken on last iteration of inner loop
- Then mispredict as not taken on first iteration of inner loop next time around

Branch Prediction State Diagram



Prediction by Branch History table