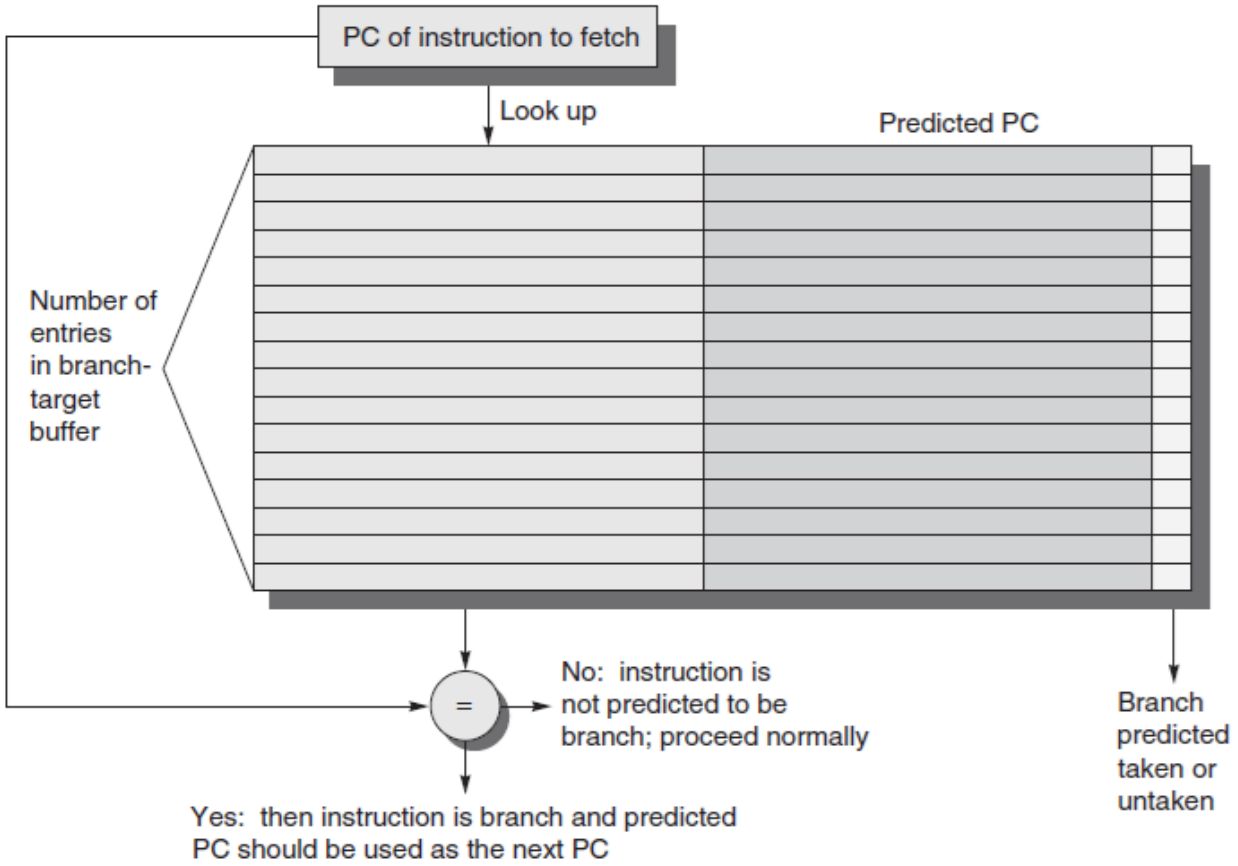
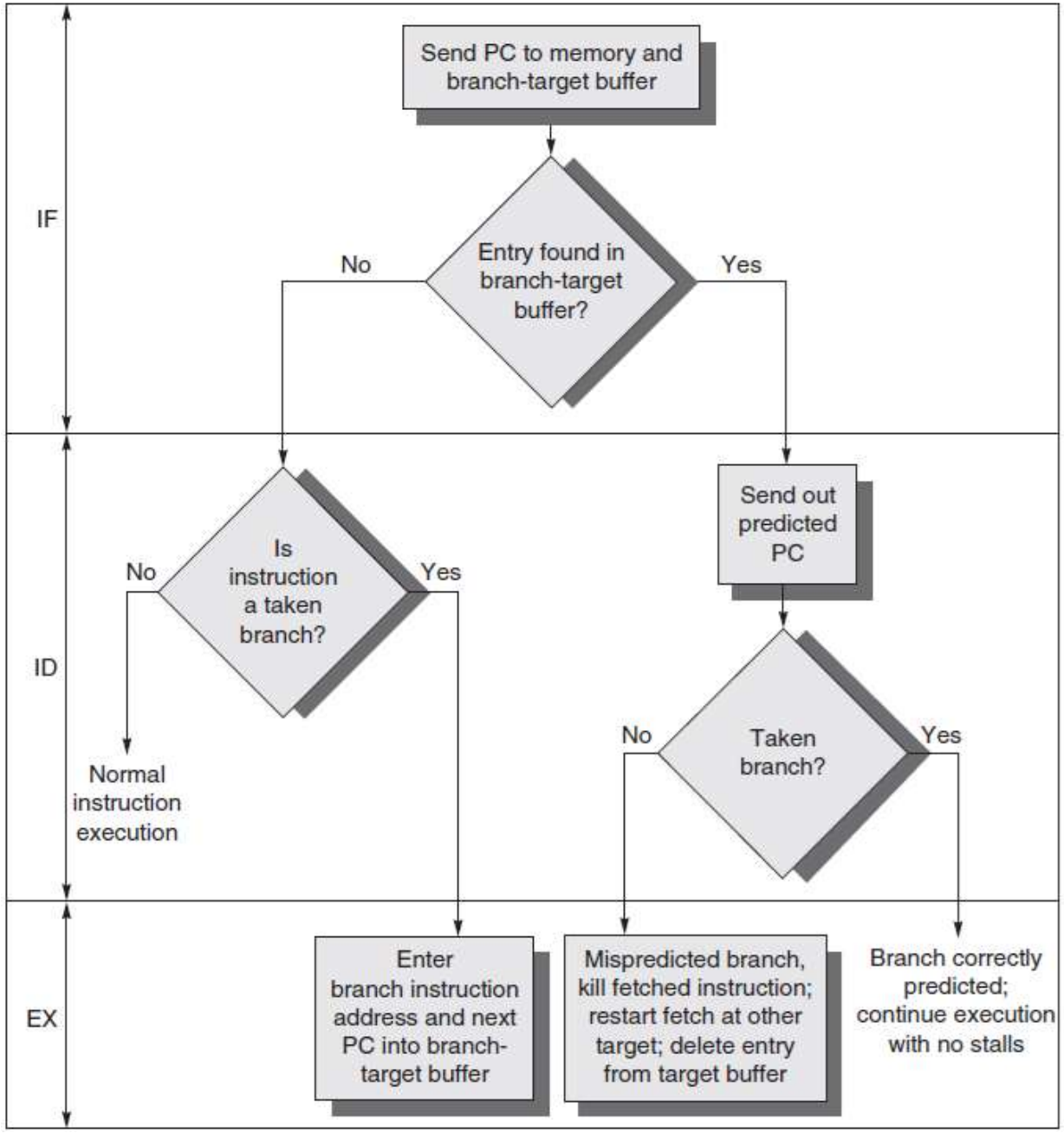
Branch Target Buffer



- If the instruction is a branch and we know what the next PC should be, we can have a branch penalty of zero.
- A branch-prediction cache that stores the predicted address for the next instruction after a branch is called a branch-target buffer or branch-target cache.

Steps using the Branch Target Buffer



Branch Prediction & Branch Target Buffer

Instruction in buffer	Prediction	Actual branch	Penalty cycles
Yes	Taken	Taken	0
Yes	Taken	Not taken	2
No		Taken	2
No		Not taken	0

- There is no branch penalty if everything is correctly predicted and the branch is found in the target buffer.
- If the branch is not correctly predicted, the penalty is equal to one clock cycle to update the buffer with the correct information (during which an instruction cannot be fetched) and one clock cycle, if needed, to restart fetching the next correct instruction for the branch.
- If the branch is not found and taken, a two-cycle penalty is encountered, during which time the buffer is updated.

Example

Instruction in buffer	Prediction	Actual branch	Penalty cycles
Yes	Taken	Taken	0
Yes	Taken	Not taken	2
No		Taken	2
No		Not taken	0

- Determine the total branch penalty for a branch-target buffer.
  - Prediction accuracy is 90% (for instructions in the buffer).
  - Hit rate in the buffer is 90% (for branches predicted taken).
- Answer
  - Probability (branch in buffer, but actually not taken)
    = Percent buffer hit rate × Percent incorrect predictions
    = 90% × 10% = 0.09
  - Probability (branch not in buffer, but actually taken) = 10%
  - Branch penalty = (0.09 + 0.10) × 2 = 0.38

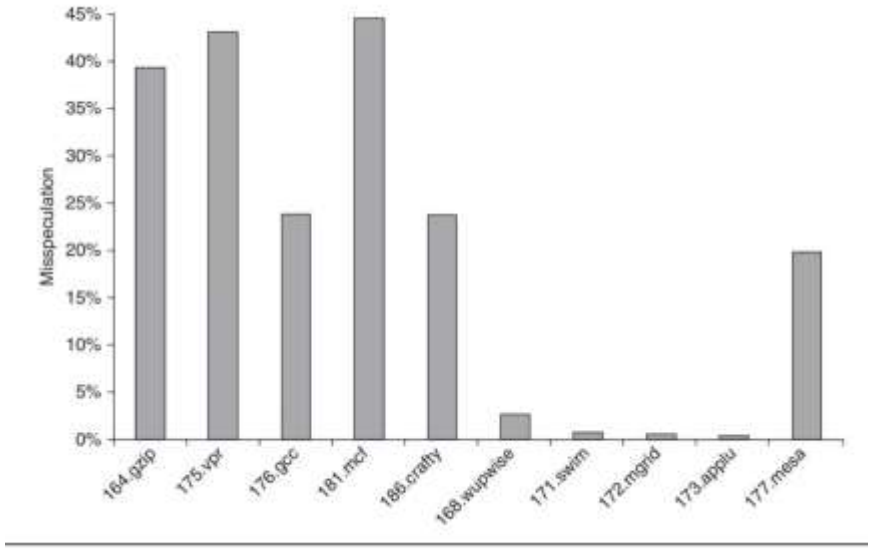
Speculation Costs and Benefits

Advantages

- Uncovers events that would stall the pipeline early (e.g., cache misses)
- Increases available instruction-level parallelism
- Allows execution to proceed past control dependencies

Costs

- Takes time and energy to execute speculative instructions
- Recovery from incorrect speculation reduces performance
- Requires additional processor resources (silicon area and power)
- Exceptional events during speculation can cause significant performance loss



The graph shows the fraction of instructions executed due to misspeculation - typically much higher for integer programs (~30%) than for FP programs, making speculation less energy efficient for integer applications.