Computer Organization & Architecture

Branch prediction

- Making branch decision in cycle 2 of execution of a branch instruction reduces the branch penalty.
  - But, instruction immediately following branch instruction is still fetched in cycle 2 and may have to be discarded.
- Decision to fetch this instruction is actually made in cycle 1:
  - When PC is incremented while branch instruction is being fetched.
- To reduce branch penalty further:
  - Processor needs to anticipate that instruction being fetched is a branch instruction
  - And predict its outcome to determine which instruction should be fetched in cycle2.

Static branch prediction

- Simplest form of branch prediction:
  - Assume that branch will not be taken
  - Fetch the next instruction in sequential address order.
  - If prediction is correct, fetched instruction is allowed to complete and there is no penalty.
- If it is determined that the branch is to be taken:
  - Instruction that has been fetched is discarded and correct branch target instruction is fetched.
  - Misprediction incurs full branch penalty.
- Same choice (assume not-taken) is used every time a conditional branch is encountered.

Static branch prediction

- If branch outcomes were random:
  - Half of all conditional branches would be taken.
  - Always assuming that branches will not be taken results in a prediction accuracy of 50 percent.
- Backward branch at end of a loop is taken most of the time.
  - For such a branch, better accuracy can be achieved by predicting that branch is likely to be taken.
  - Instructions are fetched using branch target address as soon as it is known.

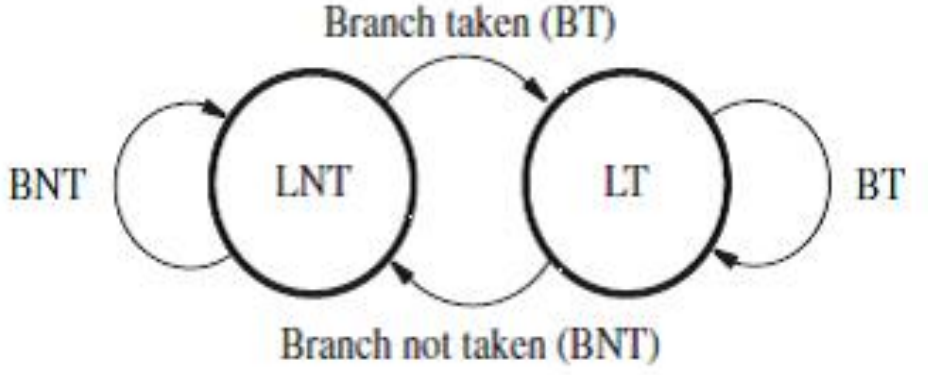
Static branch prediction

- For a forward branch at beginning of a loop, not-taken prediction leads to good prediction accuracy.
- Processor can determine static prediction of taken or not-taken:
  - By checking sign of branch offset.
  - Alternatively, machine encoding of branch instruction may include one bit that indicates whether branch should be predicted as taken or nor taken.
  - Setting of this bit can be specified by compiler.

Dynamic branch prediction

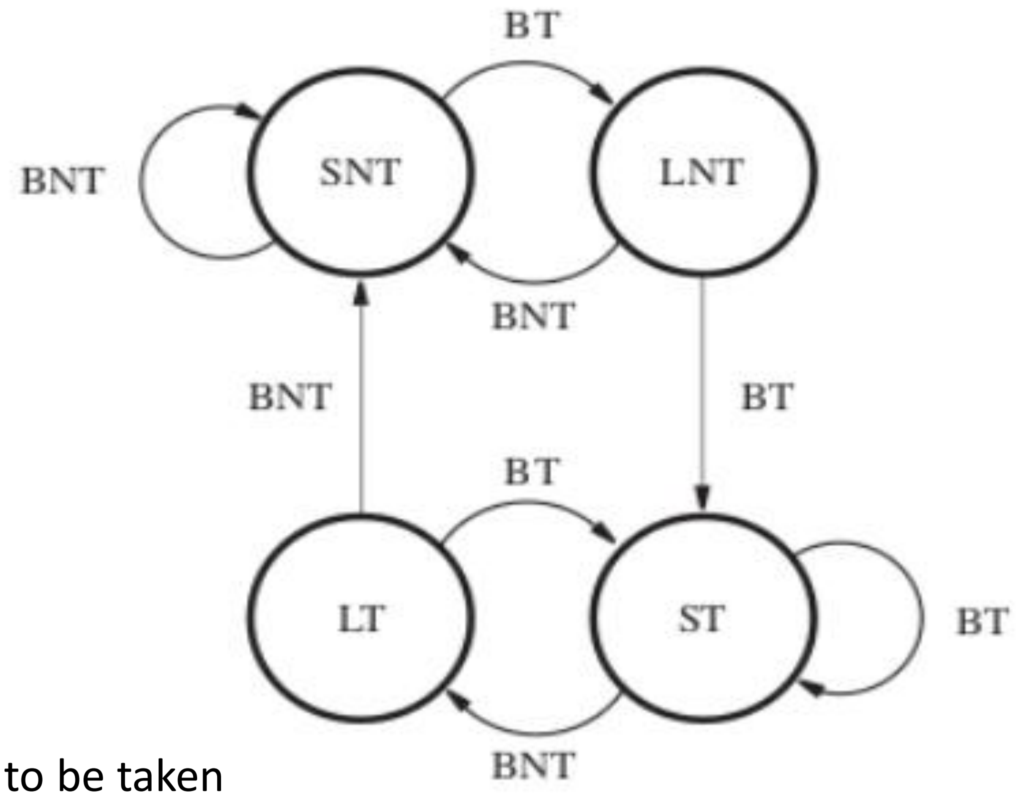
- To improve prediction accuracy further:
  - Can use actual branch behavior to influence prediction
- Processor hardware assesses likelihood of a given branch being taken:
  - By keeping track of branch decisions every time that a branch instruction is executed.
- In its simplest form, dynamic prediction algorithm can use result of most recent execution of branch instruction.
  - Processor assumes that next time instruction is executed, branch decision is likely to be same as last time.

2-state



LNT: Likely not to be taken
LT: Likely to be taken

Four-state



ST: Strongly likely to be taken
LT: Likely to be taken
LNT: Likely not to be taken
SNT: Strongly likely not to be taken

Structural hazards: Resource limitations

- Pipelining enables overlapped execution of instructions
  - But pipeline stalls when there are insufficient hardware resources to permit all actions to proceed concurrently.
- If two instructions need to access same resource in same clock cycle, one instruction must be stalled to allow other instruction to use resource.
  - Can be prevented by providing additional hardware.

Structural hazards: Resource limitations

- Ex, such stalls can occur in a computer that has a single cache that supports only one access per cycle.
  - If both Fetch and Memory stages of pipeline are connected to cache, it is not possible for activity in both stages to proceed simultaneously.
  - Normally, Fetch stage accesses the cache in every cycle.
  - This activity must be stalled for one cycle when there is a Load/Store instruction in Memory stage
  - If 25% of all instructions executed are Load or Store instructions, these stalls increase execution time by 25%
- Using separate caches for instructions and data:
  - Allows Fetch and Memory stages to proceed simultaneously without stalling.

END