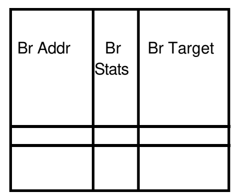
Branch Target Capture

Keeping a table of branch targets pays off in general.

The idea is to add early recognition of a branch to the predictive capabilities of branch prediction.

Branch Target Buffer

Keep a cache of recent branch instructions, their branch stats and the target address.

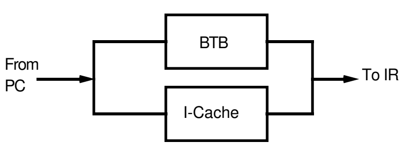


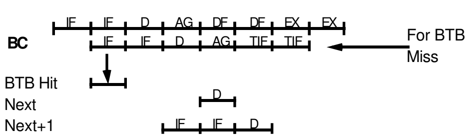
Hit Ratio, LRU

n	Compiler	Business	Scientific	OS
1	0.031	0.121	0.158	0.012
2	0.075	0.150	0.223	0.066
4	0.185	0.212	0.272	0.154
8	0.298	0.247	0.402	0.259
16	0.369	0.315	0.636	0.275
32	0.514	0.412	0.693	0.301
64	0.634	0.624	0.812	0.435
128	0.769	0.832	0.953	0.545
256	0.888	0.929	0.970	0.615
512	0.967	0.968	0.993	0.672

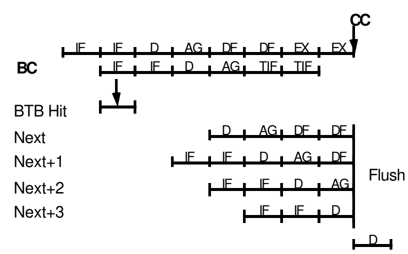
BTB Eliminates Stalls

When the BTB "hits" and the branch prediction is correct the BTB yields the instruction (or the address, since there is time to fetch it), and the decode takes place with no stall.





When the Prediction Fails

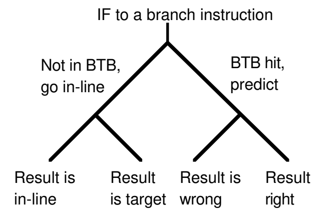


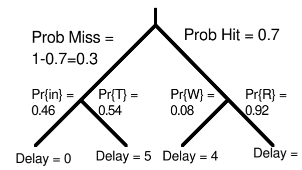
Housekeeping ...

- Must verify that the saved target didn't change.
- Must update the predictor in the table with either outcome.
- If the prediction changes, must update the target.
- If BTB miss, must add entry using some policy (LRU)

Br Addr	Br Stats	Br Target

Evaluating the Outcome





Assume the probability of being right in the prediction is 0.92

Expected Delay = 0.3*0.46*0+0.3*0.54*5+0.7*0.08*4+0.7*0.92*0 = 1.03 cycles

Guessing "always in-line" yields an expected delay of
.46*0 + .54*5 = 2.7
justifying the use of a branch target buffer.