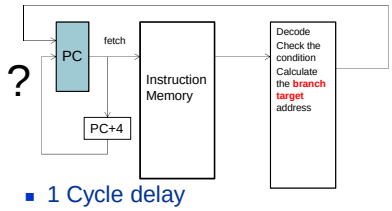
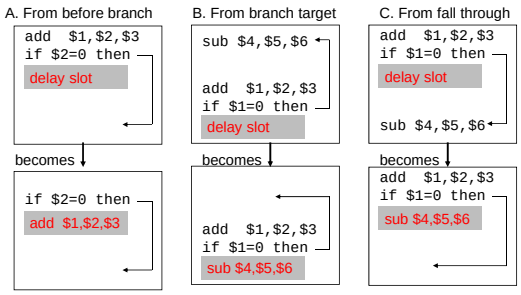
Pipeline no Prediction

- Branching completes in 2 cycles – We know the target address after the second stage



- 1 Cycle delay

Branch Delay Slots



Branch Prediction

- Dynamic scheduling deals with data dependence improving, the limiting factor is the control dependence.
- Branch prediction is important for processors that maintains a CPI of 1, but it is crucial for processors who tries to issue more than one instruction per cycle (CPI < 1).
- We have already studied some techniques (delayed branch, predict not taken), but these do not depend on the dynamic behavior of the code.

Very Simple Predictor – not practical

- Assume we have a for loop
- for(i=0; i<N; i++) in assemlbly

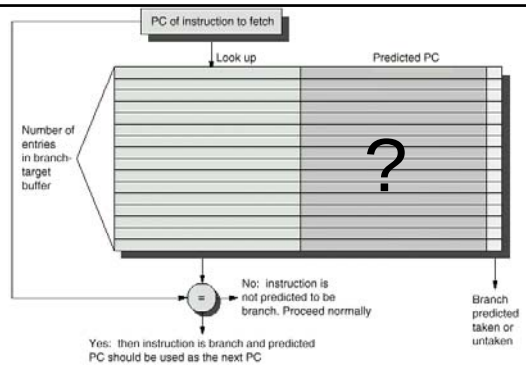
```
loop_start:     loop starts here


PC1:            beq r1, r2, loop_start
```


Branch History Table

- A small memory indexed by the lower portion of the address of the branch instruction.
- The memory contains only 1-bit, to predict taken or untaken
- If the prediction is incorrect, the prediction bit is inverted.
- In a loop, it mispredicts twice
  - End of loop case, when it exits instead of looping as before
  - First time through loop on *next* time through code, when it predicts exit instead of looping

1-Bit Predictor

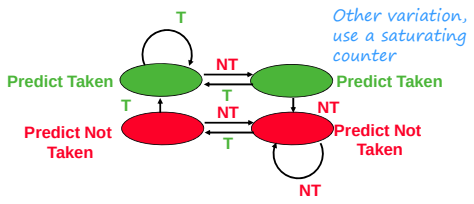
- 1-Bit bimodal predictor
- Consider the following example
- `for(i=0;i<10;i++) {`
- `……`
- `}`



Consider a loop that is taken 9 times in a row then not taken

2-Bit Predictor

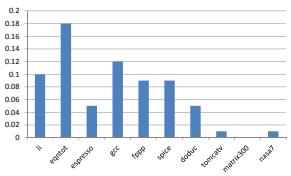
- Uses 2 bits to add some hysteresis to the prediction – Compare with 1 bit?
- 2 bits are as good as N bits (approx.)



Other variation, use a saturating counter

2-bit Predictor

- 4096 entries 2-bit predictor miss rate



Correlating Branch Predictors

```
B1    if (aa==2)              DSUBUI   R3, R1, #2
        aa=0;                 BNEZ     R3, L1      ; b1  (aa!=2)
                              DADD     R1, R0, R0  ; aa==0
B2    if (bb==2)        L1:   DSUBUI   R3, R1, #2
        bb=0;                 BNEZ     R3, L2      ; b2  (bb!=2)
                              DADD     R2, R0, R0  ; bb==0
B3    if (aa!==bb){     L2:   DSUBUI   R3, R1, R2  ; R3=aa-bb
                              BEQZ     R3, L3      ; b3  (aa==bb)
```

If the condition is true → (B1,B2) branch NOT TAKEN

If the condition is true → B3 NOT taken

If B1 and B2 both NOT TAKEN B3 → TAKEN

There is a correlation between B3 and both B1 and B2

Correlating Branch Predictors

- Correlating predictors (two-level predictors) use the behavior of other branches to make prediction.
- Simplest (1-bit) has 2 predictions, one if the last branch is take, the second is when the last branch is not taken
- The prediction is on the form NT/T

Example

```
B1    if (d==0)
        d=1;
B2    if (d==1)
        {
```

```
      BNEZ     R1, L1      ; d == 0 ?
      DADD     R1, R0, #1  ; YES d==1
L1:   DADD     R3, R1, #-1
      BNEZ     R3, L2      ; b2  (bb!=2)

L2:
```

If b1 not taken, b2 is taken for sure

Initial d	d==0?	B1	d befoe b2	d==1	b2
0	Y	NO	1	Y	NO
1	N	Taken	1	Y	NO
2	N	Taken	2	N	Taken

Example

Initial d	d==0?	B1
0	Y	NO
1	N	Taken
2	N	Taken

d befoe b2	d==1	B2
1	Y	NO
1	Y	NO
2	N	Taken

d	B1 Pred	B1 action	newB1 pred	B2 pred	B2 action	new B2 pred
2	NT	T	T	NT	T	T
0	T	NT	NT	T	NT	NT
2	NT	T	T	NT	T	T
0	T	NT	NT	T	NT	NT

Miss on every prediction

Example

Initial d	d==0?	B1
0	Y	NO
1	N	Taken
2	N	Taken

d befoe b2	d==1	b2
1	Y	NO
1	Y	NO
2	N	Taken

d	b1 Pred		b1 action	newb1 pred	b2 pred		b2 action	new b2 pred
2	NT/NT	%	T	T/NT	NT/NT	%	T	NT/T
0	T/NT	√	NT	T/NT	NT/T	√	NT	NT/T
2	T/NT	√	T	T/NT	NT/T	√	T	NT/T
0	T /NT	√	NT	T/NT	NT/T	√	NT	NT/T

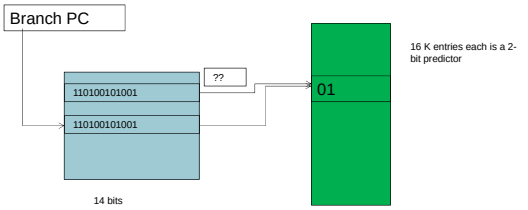
Misprediction on first try only

Global Predictor

- Take for example 10 bits of the branch PC
- Take 4 bits of global branch history
- Access 2^{14} entry table
- Or, you could take the 14 bits of PC XORED with 14 bits of branch history (hashing) to access the same table
- Or any combination

Local Predictor

- The recent history of the branch predicts the next one



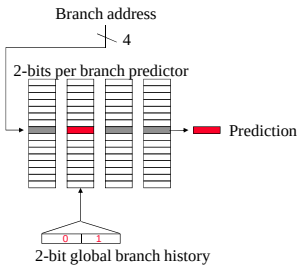
Correlating Predictors

- The 1-bit predictor is called (1,1) predictor.
- It uses one bit for history (last branch), to choose among two (2^1) 1-bit branch predictors.
- In general a predictor could me *(m,n)* predictor.
- It uses the last m branch to choose among 2^m branch predictors each is n-bit predictor.

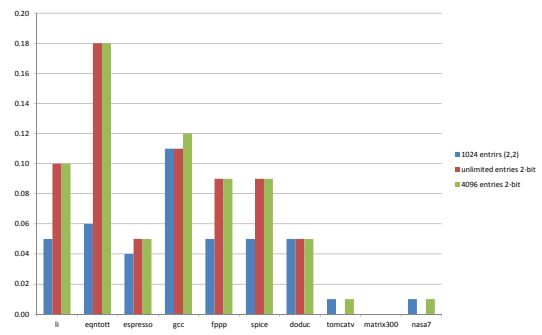
(2,2) Correlating Predictors

(2,2) predictor

– Behavior of recent branches selects between four predictions of next branch, updating just that prediction



Comparison

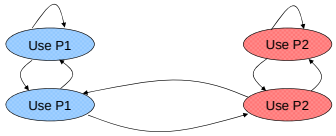


Branch Prediction

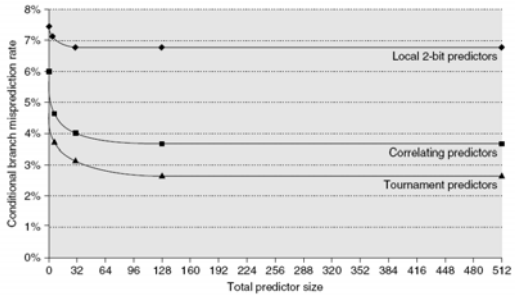
- Basic 2-bit predictor:
  - For each branch:
    - Predict taken or not taken
    - If the prediction is wrong two consecutive times, change prediction
- Correlating predictor:
  - Multiple 2-bit predictors for each branch
  - One for each possible combination of outcomes of preceding *n* branches
- Local predictor:
  - Multiple 2-bit predictors for each branch
  - One for each possible combination of outcomes for the last *n* occurrences of this branch

Tournament Predictor

- Tournament predictor:
  - Combine correlating predictor with local predictor
  - A selector is sued to decide which one of these to use
- The selector could be similar to a 2-bit predictor
  - A saturating 2-bit binary counter with 2 outcomes P1/P2



Branch Prediction Performance



Branch predictor performance

Alpha 21264 Branch Predictor

- Tournament predictor using, 4K 2-bit counters indexed by local branch address.
- Global predictor
  - 4K entries index by history of last 12 branches (2^{12} = 4K)
  - Each entry is a standard 2-bit predictor
- Local predictor
  - Local history table: 1024 10-bit entries recording last 10 branches, index by branch address
  - The pattern of the last 10 occurrences of that **particular** branch used to index table of 1K entries with 3-bit saturating counters

Branch Target Buffer

- Prediction tells us if the branch is taken or not.
- If taken, to where? Target address
- Branch target buffer tells us where (based on the PC, or parts of the PC).