ECE 4750 Computer Architecture, Fall 2015

T13 Advanced Processors: Branch Prediction

School of Electrical and Computer Engineering
Cornell University

revision: 2015-11-13-14-20

1. Branch Prediction Overview

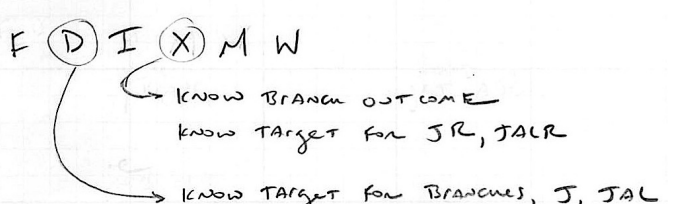
branch												
opA												
opB												
opC												
opD												
opE												
opF												
opG												
opH												

ESSENTIAL IN MODERN PROCESSORS TO AVOID MITIGATE BRANCH DELAY LATENCIES.

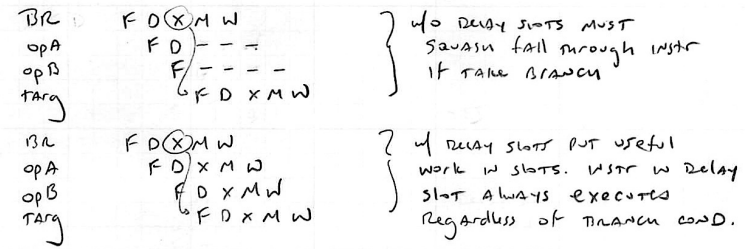
2 KEY TYPES OF PREDICTION

1. PREDICT BRANCH OUTCOME
2. PREDICT BRANCH | JUMP TARGET ADDRESS

PARC PIPELINE



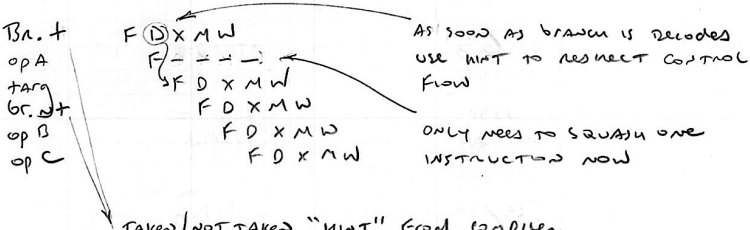
SW BPRED: BRANCH DELAY SLOTS



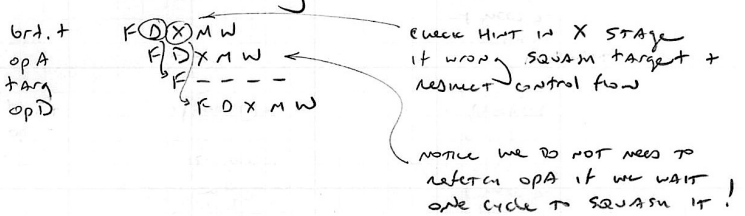
EXPOSES TOO MUCH ABOUT MICROARCH, LESS POPULAR NOW

SW BPRED: STATIC SW PREDICTION

EXTEND ISA SO THAT COMPILER CAN TELL MICROARCHITECTURE IF BRANCH IS LIKELY TO BE TAKEN OR NOT



WHAT IF HINT IS WRONG?



STATIC SW PREDICTION CAN WORK BECAUSE WE CAN EXPLOIT STATIC STRUCTURE IN THE PROGRAM.

FOR EXAMPLE, MOST BACKWARDS BRANCHES ARE TAKEN SO USE HIT TO SPECIFY TAKEN FOR THESE BRANCHES.

SW BPRED : PREDICATION

NOT really "PREDICTION". IDEA IS TO TURN CONTROL FLOW INTO DATAFLOW COMPLETELY ELIMINATING CONTROL HAZARD

```
MOVN rd, rs, rt     if (R[rt] != 0) R[rd] ← R[rs]
MOVZ rd, rs, rt     if (R[rt] == 0) R[rd] ← R[rs]
```

PSEUDOCODE	w/o PREDICATION	w/ PREDICATION

if (a < b)
  x = a
else
  x = b
	slt r1, r2, r3
	beqz r1, L1
	move r4, r2
	j L2
	L1:
	move r4, r3
	L2:
		slt r1, r2, r3
		movz r4, r2, r1
		movn r4, r3, r1

WHAT IF /THEN/ELSE HAS MANY INSTRUCTIONS?

WHAT IF ONE SIDE OF THE BRANCH HAS MANY MORE INSTRUCTIONS THAN THE OTHER SIDE?

FULL PREDICATION

- ALMOST ALL INSTRUCTIONS CAN be EXECUTED UNDER PREDICATE
- INSTRUCTION BECOMES NOP IF PREDICATE IS FALSE

```
if (a < b)          slt.p p1, r2, r3    ← special predicate flag
  op A        (p1)  op A
  op B        (p1)  op B
else          (!p1) op C
  op C        (!p1) op D
  op D
```

HW BPRED: FIXED PREDICTION

1. ALWAYS PREDICT NOT-TAKEN
   - WHAT WE HAVE BEEN ASSUMING
   - SIMPLE TO IMPLEMENT
   - KNOW FALL THROUGH PC IN F
   - POOR ACCURACY, ESPECIALLY ON VERY IMPORTANT BACKWARDS BRANCH IN LOOPS.

2. ALWAYS PREDICT TAKEN
   - DIFFICULT TO IMPLEMENT BECAUSE DON'T KNOW TARGET UNTIL D
   - COULD STILL RESOLVE BRANCH DELAY LATENCY IN DARE BY ONE
   - POOR ACCURACY, ESPECIALLY ON IF-THEN-ELSE

3. BACKWARD BRANCH TAKEN, FORWARD BRANCH NOT TAKEN
   - SIMILAR TO (2) IN TERMS OF TARGET
   - BETTER ACCURACY

```
loop:
    LW    r1, 0(r2)
    LW    r3, 0(r4)
    SLT   r5, r1, r3
    BEQZ  r5, L1
    MOVE  r6, r1
    J     L2
L1: MOVE  r6, r3
L2: SW    r6, 0(r7)
    ADDIU r2, r2, 4
    ADDIU r4, r4, 4
    ADDIU r7, r7, 4
    ADDIU r8, r8, -1
    BGTZ  r8, loop
```

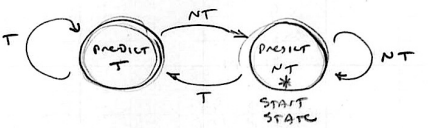
FORWARD BRANCH ON AVG 50% TAKEN

BACKWARDS BRANCH ON AVG 90% TAKEN

HW BPRED: EXPLOITING TEMPORAL CORRELATION

* EXPLOIT STRUCTURE IN PROGRAM: THE WAY A BRANCH RESOLVES MAY BE A GOOD INDICATION OF THE WAY IT WILL RESOLVE THE NEXT TIME IT IS EXECUTED (TEMPORAL CORRELATION)

1-BIT SATURATING COUNTER



Remember condition of LAST branch execution + predict it goes same way

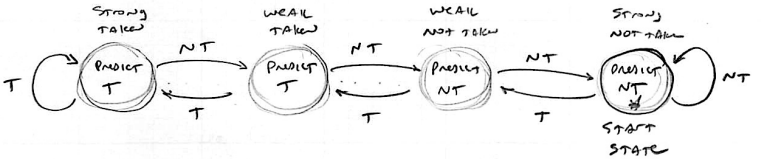
ITERATION	PREDICTION	ACTUAL	MISPREDICT.?
1	NT	T	Y
2	T	T	
3	T	T	
4	T	NT	Y
⋮			
1	NT	T	Y
2	T	T	
3	T	T	
4	T	NT	Y

for backwards branch in loop

- ASSUME 4 iterations
- ASSUME loop is executed multiple times

ALWAYS 2 MISPREDICTS. LOOPS ARE VERY COMMON
EXPLOITING TEMPORAL CORRELATION WORKS VERY WELL

2-bit saturating counter

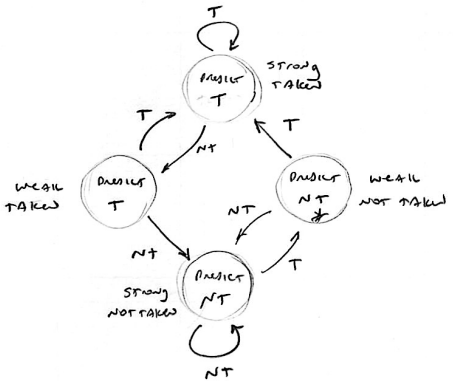


Iteration	Prediction	Actual	Mispredict?	State
1	NT	T	Y	strong NT
2	NT	T	Y	weak NT
3	T	T		weak T
4	T	NT		strong T
⋮				
1	T	T		strong T
2	T	T		strong T
3	T	T		strong T
4	T	NT	Y	weak T

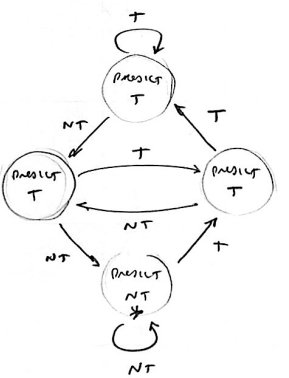
What if start state is strong taken?

only 1 mispredict per loop.

OTHER 2-BIT FSM BRANCH PREDICTORS



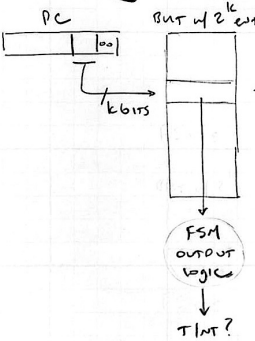
JUMP DIRECTLY TO STRONG FROM WEAK



BIASED TOWARDS PREDICTING BRANCHES AS TAKEN

SEE FIG 5.8 IN SUEN + LIPASTI FOR OTHER ALTERNATIVES

Implementing FSMs with Branch History Table



FSM state for specific PC

Two PC's can "alias" to the same entry in BHT. Similar to a cache conflict, except we can't really tell when a conflict happens

Storing the PC as a tag is too expensive

Aliasing = ↑ mispredictions

↓ aliasing w/ larger BHT or can also ↑ associativity (still only use a few bits as "tag" in associative structure.)

Most BHTs are direct mapped. Because with only 2-bits per entry cheap to use large BHT

* 4k 2bits/entry BHT
  = 80-90% prediction accuracy

How do we continue to improve prediction accuracy? Exploit more sophisticated temporal correlation

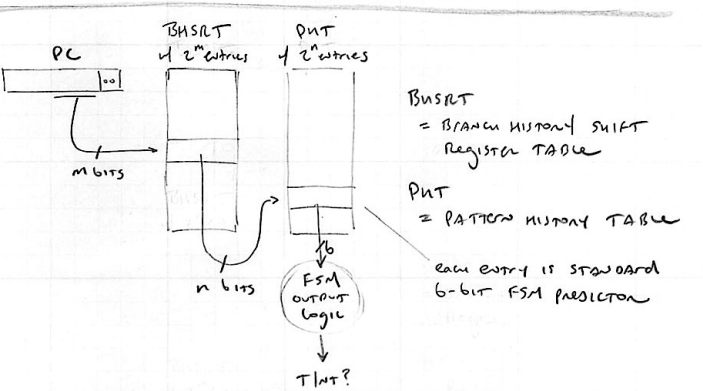
More Complicated Temporal Correlation

Often a branch exhibits more complicated pattern than just "always taken" or "always not taken". Could develop a more complicated FSM, but these patterns vary per branch. We want per branch customized FSMs.

```
void Convolve( int B[], int A[], int size ) {
    for (int i = 2; i < size-2; i++)
        for (int j=0; j < 5; j++)
            B[i - (2-j)] = A[i] * COEFF[j]
}
```

Can we predict that every fifth dynamic instance of the backwards loop branch will be not taken?

2-Level BHT to Exploit Temporal Correlation



When a branch is taken or not taken we shift in either a one (taken) or zero (not taken) into the least significant bit of the corresponding BHSR.

The BHSR captures the temporal pattern for that branch

We use BHSR to index into the PHT. A BHT has an entry per branch, but a PHT has an entry per branch pattern.

The PHT says for a given pattern over the past n executions of a branch, should I take or not take the next execution of this branch?

So in previous example pattern for inner loop backwards branch will be

0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 ...

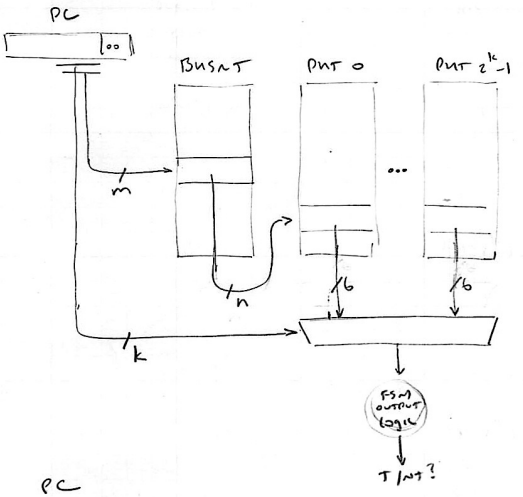
With 2-level BHT and for $n \geq 4$ we can learn this pattern and perfectly predict this branch.

Problem: Aliasing in the BHSRT

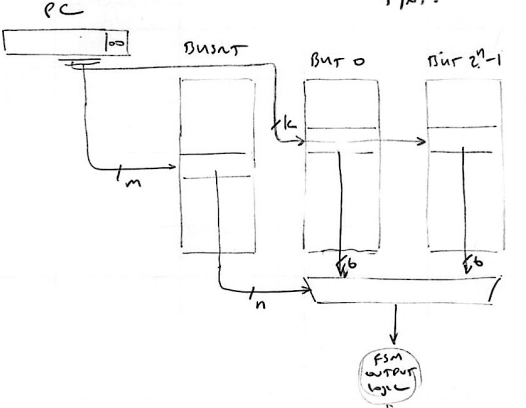
Solution: Make BHSRT larger or increase associativity

PROBLEM: MULTIPLE BRANCHES WITH SAME HISTORY MIGHT NEED DIFFERENT PREDICTIONS. IN OTHER WORDS, ALIASING IN THE PHT CAN REDUCE ACCURACY

SOLUTION: ADD MULTIPLE PHTS, USE BITS FROM PC TO CHOOSE WHICH PHT TO USE



ISOMORPHIC - TWO DIFFERENT WAYS OF DRAWING THE SAME TWO-level STRUCTURE.



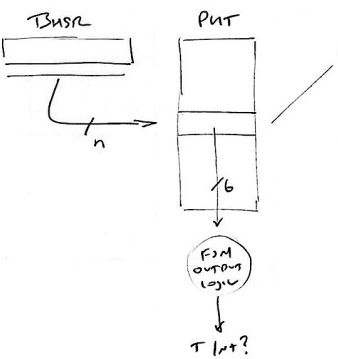
HW BPRED: EXPLOITING SPATIAL CORRELATION

THE WAY ONE BRANCH IS RESOLVED MAY BE A GOOD INDICATION OF THE WAY A LATER (DIFFERENT) BRANCH WILL RESOLVE

```
if (x < 7)                slt   r2, r1, 7
  Y++                     beq   r2, L1        branch ϕ
if (x < 5)                ADDIU r3, r3, 1
  Z++                   L1:
                          slt   r2, r1, 5
                          beq   r2, L2        branch 1
                          ADDIU r4, r4, 1
                        L2:
```

IF BRANCH ϕ IS TAKEN (IE $x \geq 7$)
THEN BRANCH 1 IS ALWAYS TAKEN (IE x MUST BE ≥ 5)

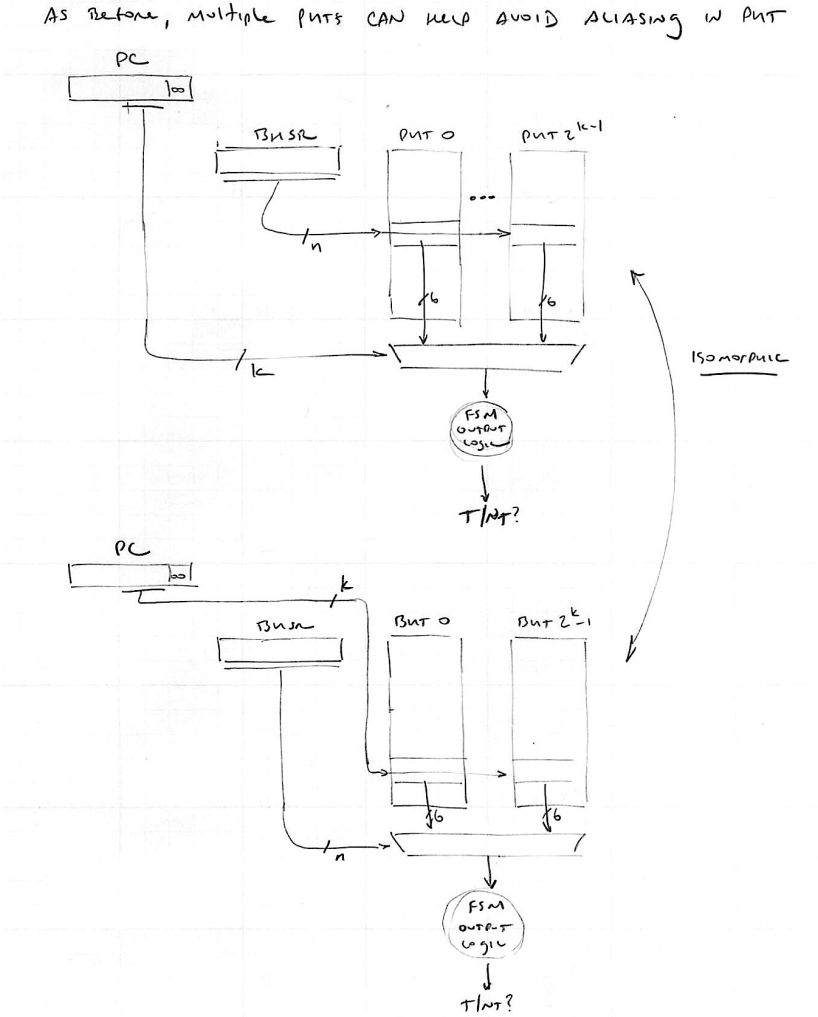
SO WHETHER BRANCH ϕ IS TAKEN OR NOT TAKEN CAN BE USED TO PREDICT IF WE SHOULD TAKE BRANCH 1



STANDARD 6-BIT FSM PREDICTOR

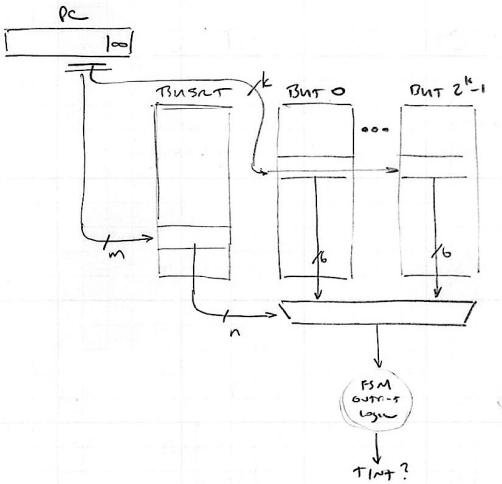
ONLY ONE BHSR, SO HISTORY OF ALL BRANCHES MERGED INTO A SINGLE HISTORY SHIFT REGISTER

FOR ABOVE EXAMPLE, BHSR WILL CAPTURE HISTORY – SO WE WILL KNOW WHEN FIRST BRANCH IS TAKEN, AND THAT VALUE (1) OF THE BHSR WILL POINT TO AN ENTRY IN THE PHT THAT PREDICTS TAKEN.


AS BEFORE, MULTIPLE PHTS CAN HELP AVOID ALIASING IN PHT
PC
BHSR
PHT 0
PHT 2^k-1
n
k
FSM OUTPUT LOGIC
T/NT?
ISOMORPHIC
PC
k
BHSR
BHT 0
BHT 2^k-1
n
FSM OUTPUT LOGIC
T/NT?

GENERALIZED TWO-LEVEL BHTS

COMBINED APPROACH TO EXPLOIT BOTH COMPLEX TEMPORAL CORRELATION AND SPATIAL CORRELATION



DIFFERENCE FROM DISCUSSION ON COMPLEX TEMPORAL CORRELATION IS THAT WE PURPOSELY CHOOSE A SMALLER m TO CAUSE ALIASING IN THE BHSR SINCE THIS ALIASING ALLOWS US TO CAPTURE SPATIAL CORRELATION

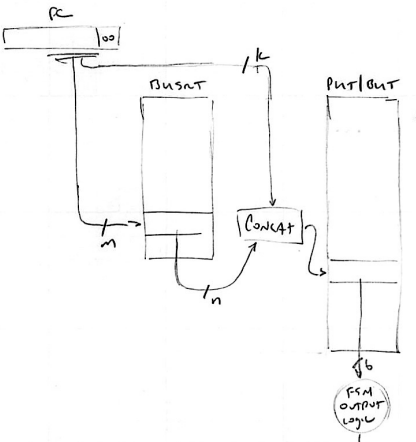
(CHOOSE HIGHER ORDER m BITS)

		ONE PHT $k=0$	$0<k<30$	PHT for each PC $k=30$
ONE BHSR	$m=0$	GAg	GAs	GAp
	$0<m<30$	PAg	PAs	PAp
ONE BHSR for each PC	$m=30$	SAg	SAs	SAp

97% ACCURACY

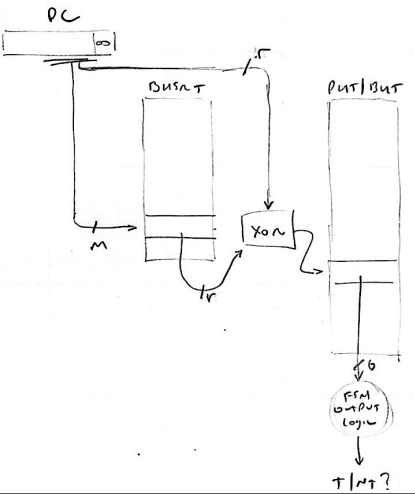
g-select

Isomorphic to previous figure



Instead of concatenating XOR bits from PC with BHSR, helps avoid aliasing in the PHT/BHT more effectively

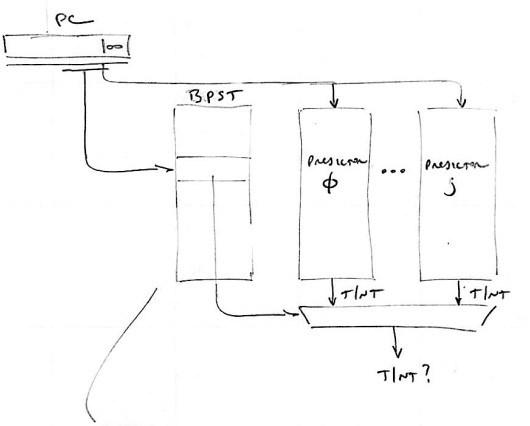
g-share



Tournament Predictors

Different predictors are better at predicting different types of branches

- one-level 2-bit saturating counter — loops
- two-level gshare — irregular code

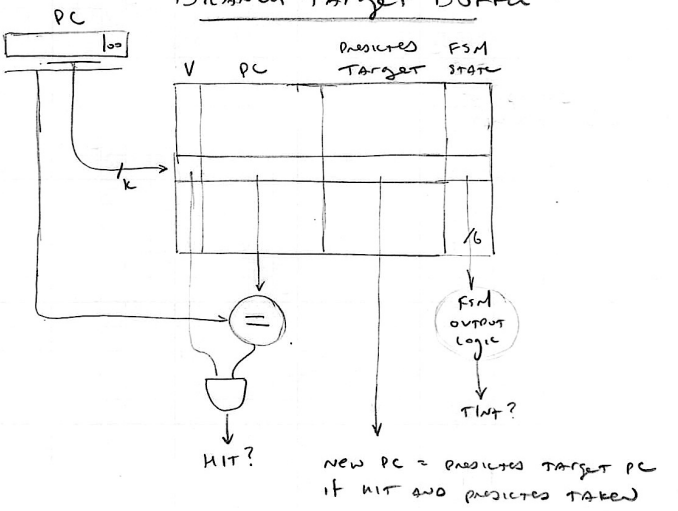


Branch predictor selector table
— predicts which branch predictor we should use

HW BPRED: PREDICTING TARGET ADDRESS

EVEN WITH BEST POSSIBLE PREDICTION OF BRANCH OUTCOME STILL NEED TO WAIT FOR TARGET ADDRESS TO BE DETERMINED.

BRANCH TARGET BUFFER

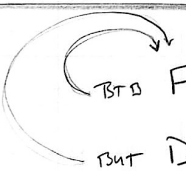


CAN PUT BTB IN FETCH STAGE

- PREDICTING IF PC POINTS TO A BRANCH
- PREDICTING TARGET OF BRANCH
- PREDICTING IF BRANCH IS TAKEN.

SOMETIMES $b=0$, IF HIT THEN ASSUME PREDICT TAKEN

COMBINE BTB AND BHT



BTB IS MUCH MORE EXPENSIVE THAN BHT, BUT BTB EARLIER IN PIPELINE + CAN ACCELERATE JR

COMBINE BTB W/ FEW ENTRIES WITH BHT W/ MANY ENTRIES

RETURN ADDRESS STACK PREDICTOR

BTB ONLY WORKS FOR JR FUNCTION CALL RETURNS IF ALWAYS CALL FUNCTION FROM SAME PLACE (NOT REALISTIC)

STACK PREDICTOR

- PUSH TARGET ADDRESS ON STACK for JAL/JALR
- POP OFF TARGET ADDRESS for JR TO PREDICT TARGET

MOVE STACK PREDICTOR WHO FETCH AND PREDICT WHICH PC'S ARE JR.

USE TOURNAMENT PREDICTOR TO CHOOSE BETWEEN BTB AND STACK PREDICTOR.