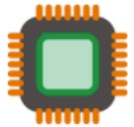


CASPER

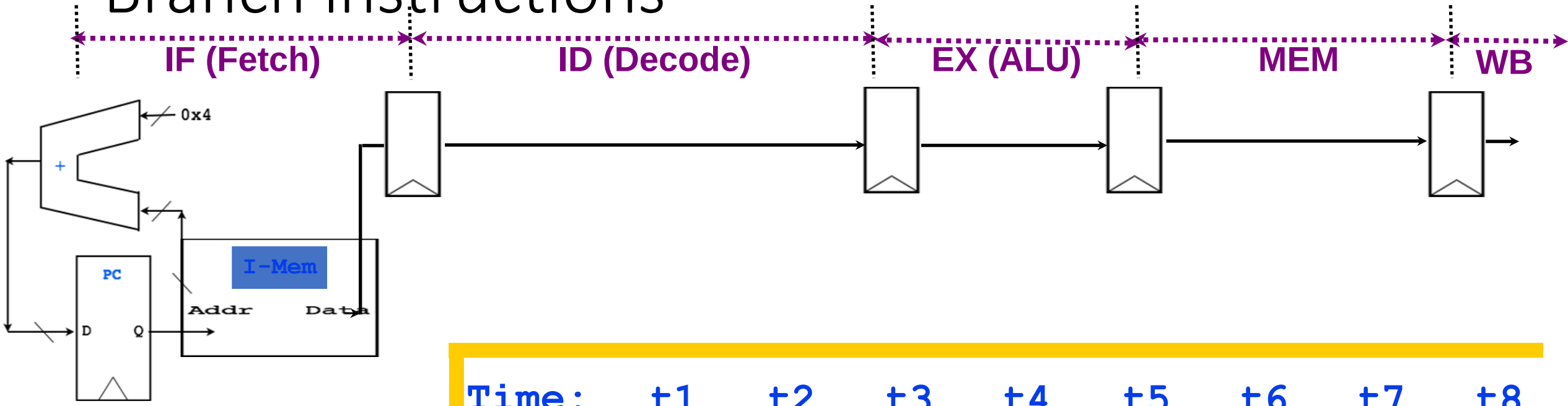
CS305: Computer Architecture

Branch Prediction

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

Branch instructions

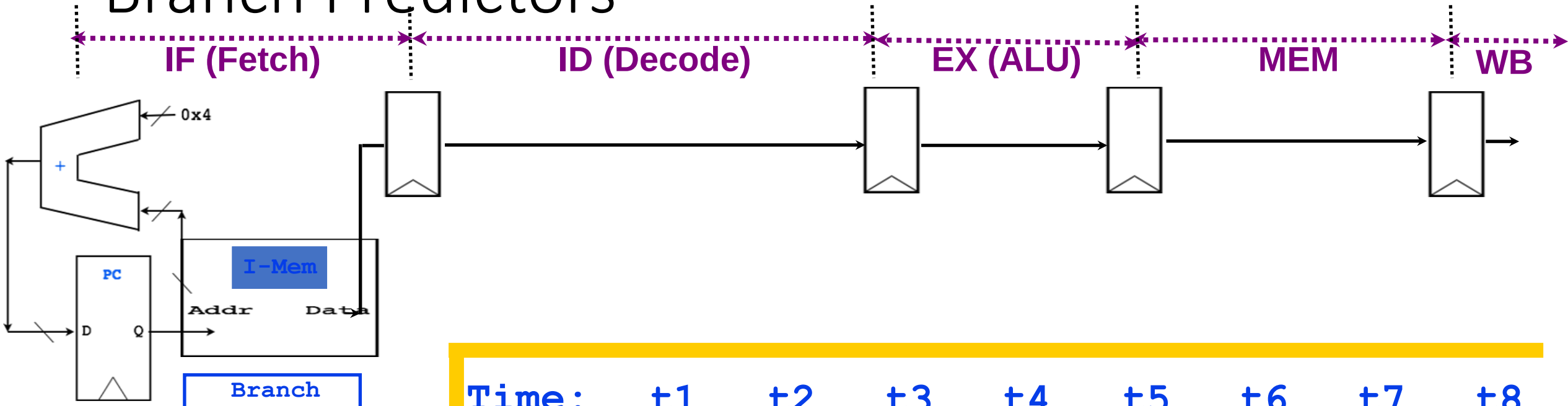


Time:	t1	t2	t3	t4	t5	t6	t7	t8
Inst								
I1:	IF	ID	EX	MEM	WB			
I2:		IF	ID					
I3:			IF					
I4:								
I5:								
I6:								

EX stage computes if branch is taken

If branch is taken, these instructions MUST NOT complete!

Branch Predictors



Time:	t1	t2	t3	t4	t5	t6	t7	t8
Inst								
I1:	IF	ID	EX	MEM	WB			
I2:		IF	ID					
I3:			IF					
I4:								
I5:								
I6:								

EX stage computes if branch is taken

If branch is taken, these instructions MUST NOT complete!

A quick recap

What if PC=PC+4? Not TRUE

Flush/kill all the instructions in the wrong path.

Branch Prediction: 10K Feet View

Predict whether the next PC is a branch PC, at the fetch stage?

Branch Prediction: 10K Feet View

Predict whether the next PC is a branch PC, at the fetch stage?

If branch, will it be taken?

Branch Prediction: 10K Feet View

Predict whether the next PC is a branch PC, at the fetch stage?

If branch, will it be taken?

If taken, what is the target address?

Branch Prediction: 10K Feet View

Predict whether the next PC is a branch PC, at the fetch stage?

If branch, will it be taken?

If taken, what is the target address?

How?

Branch Prediction: 10K Feet View

Predict whether the next PC is a branch PC, at the fetch stage?

If branch, will it be taken?

If taken, what is the target address?

How?

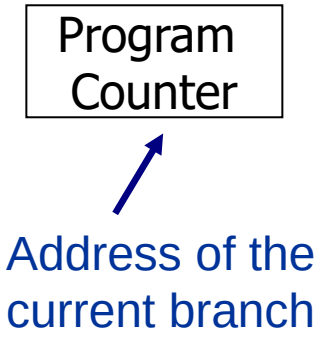
We know whether it is a branch PC or not in the decode stage. Oh no ☹

Branch Predictor: A bit deeper

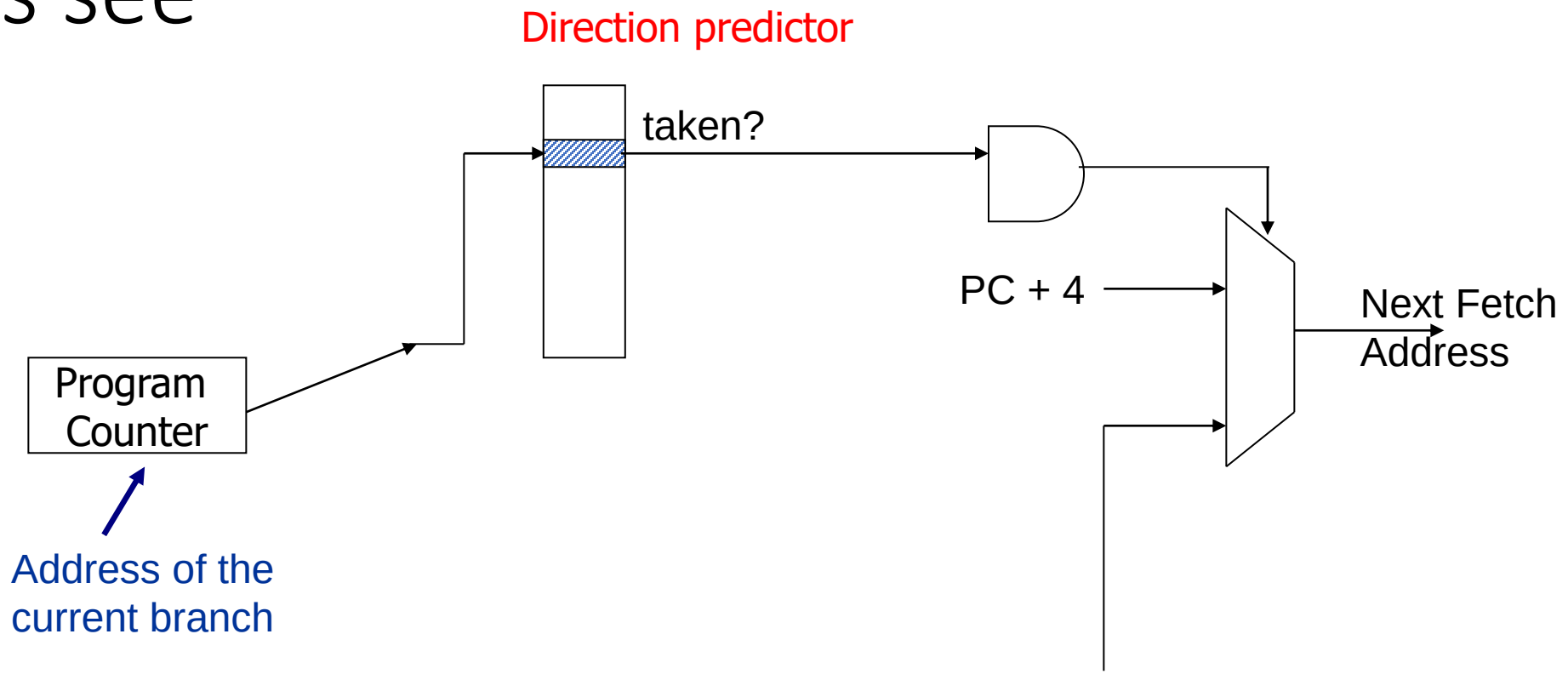
Three tasks

1. Is the PC a branch/jump? YES/NO
2. If Yes, can we predict the direction? Taken or not-taken
3. If taken, can we predict the target address?

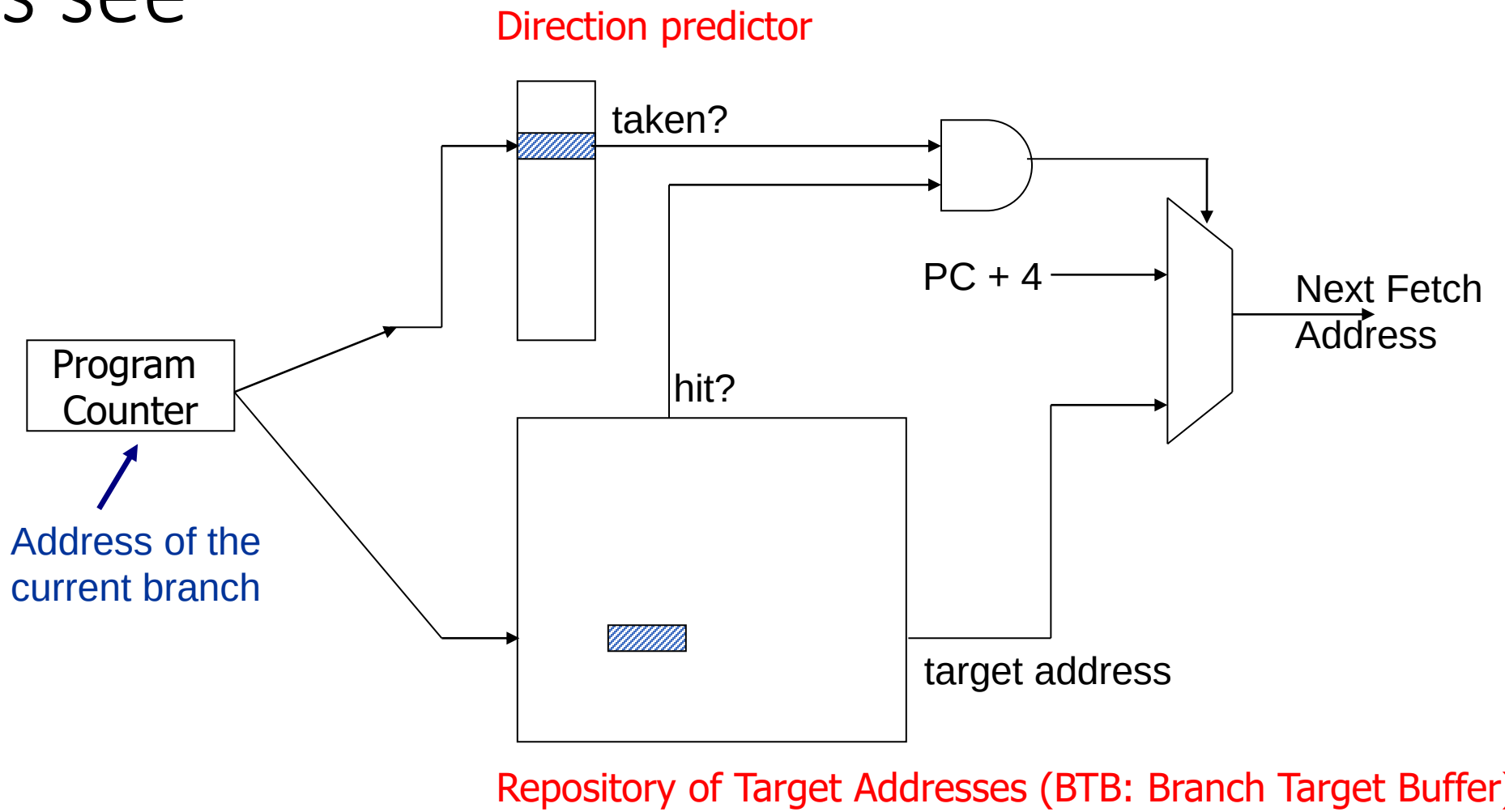
Let’s see



Let’s see


Direction predictor
taken?
PC + 4
Next Fetch
Address
Program
Counter
Address of the
current branch

Let's see


Direction predictor
taken?
PC + 4
Next Fetch Address
Program Counter
hit?
Address of the current branch
target address
Repository of Target Addresses (BTB: Branch Target Buffer)

Static (compiler) Direction Prediction Techniques

Always not-taken: Simple to implement: no need for BTB, no direction prediction

Low accuracy: ~30-40%

Always taken: No direction prediction, we need BTB though

Better accuracy: ~60-70%

Backward branches (i.e., loop branches) are usually taken

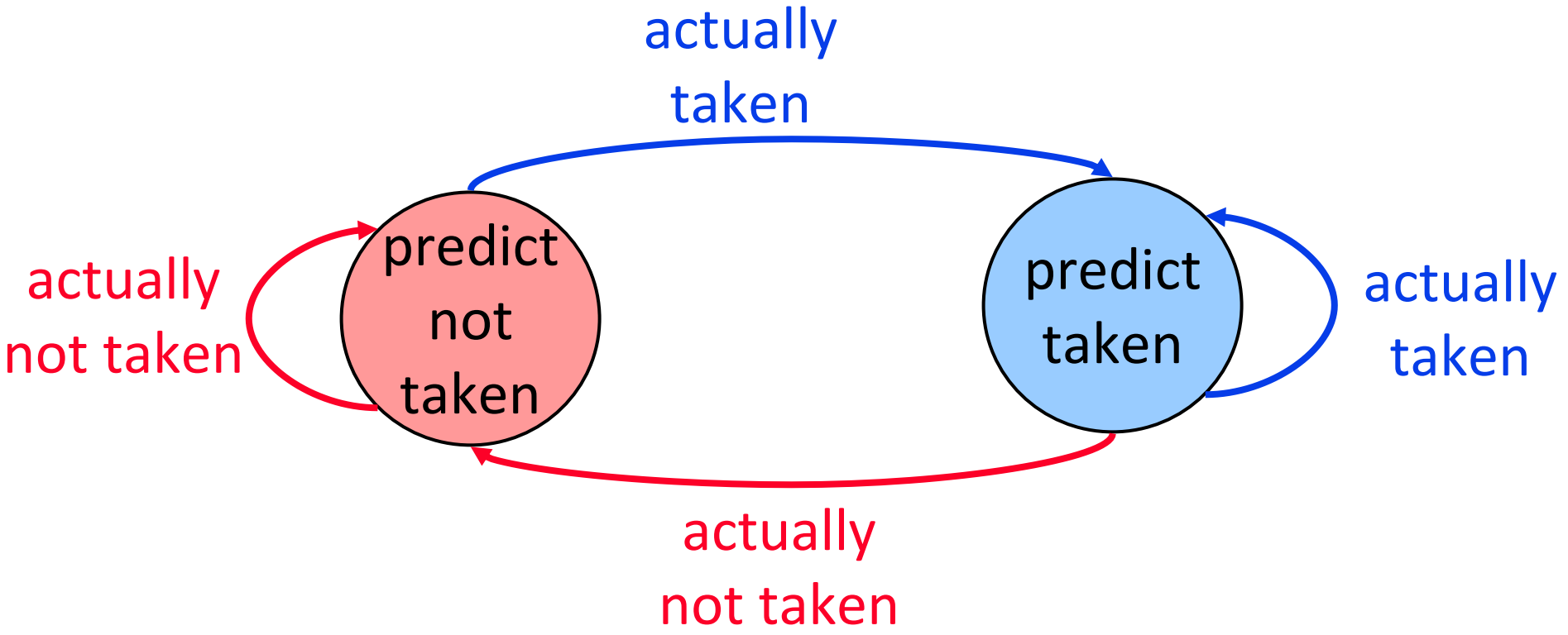
Dynamic Predictors

Microarchitectural way of predicting it.

Simple one: Last time predictor

Last-time predictor

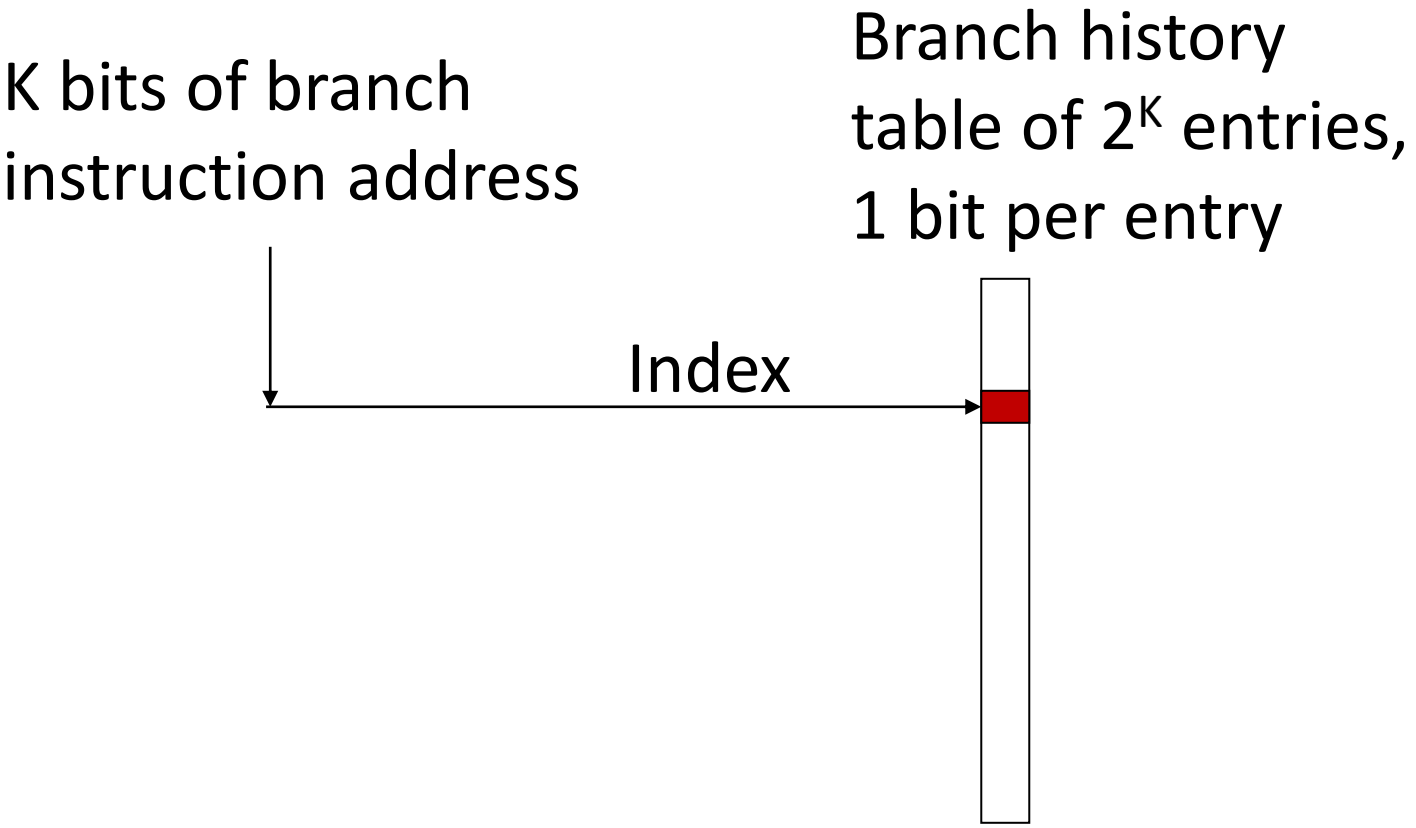
Last-time predictor



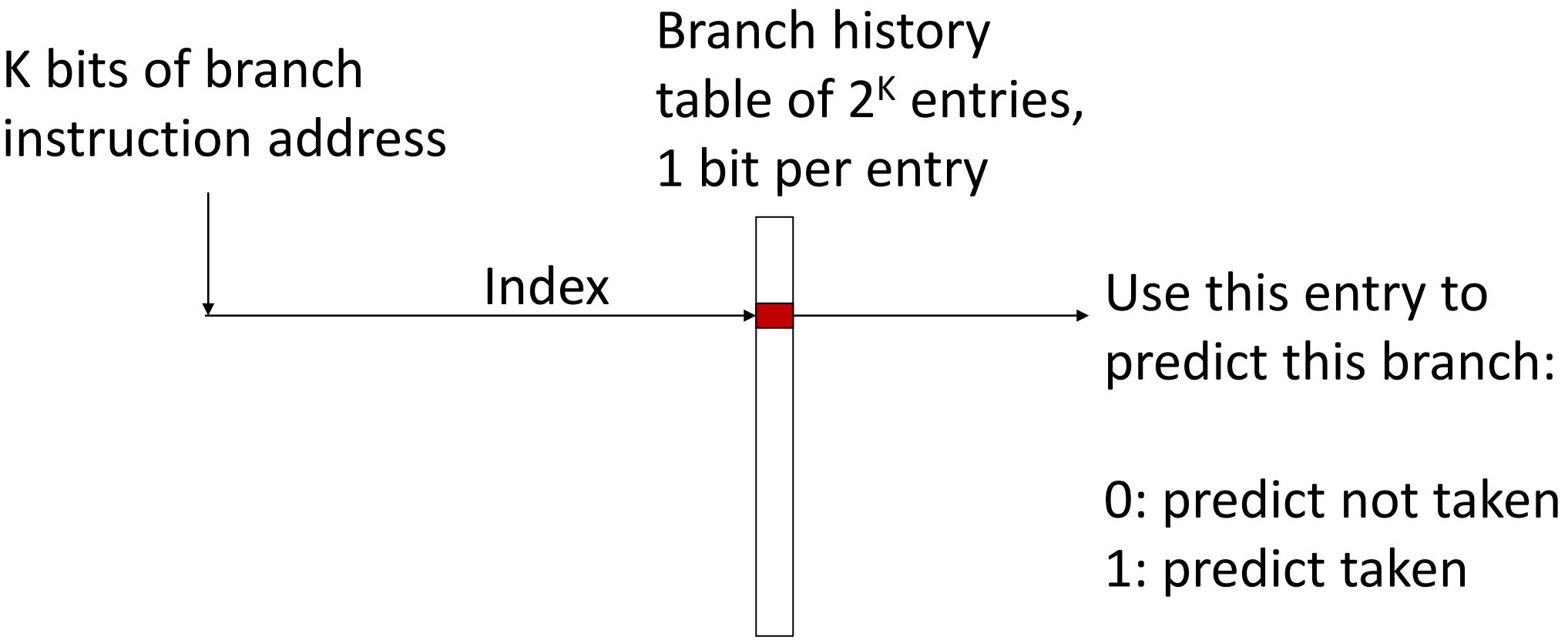
Implementation

K bits of branch instruction address

Implementation



Implementation



Performance of Last-time predictor

TTTTTTTTTTNNNNNNNNNN - 90% accuracy

Always mispredicts the last iteration and the first iteration of a loop branch

Accuracy for a loop with N iterations = (N-2)/N

+ Loop branches for loops with large number of iterations

-- Loop branches for loops will small number of iterations

Performance contd.

TNTNTNTNTNTNTNTNTNTNTN → 0% accuracy

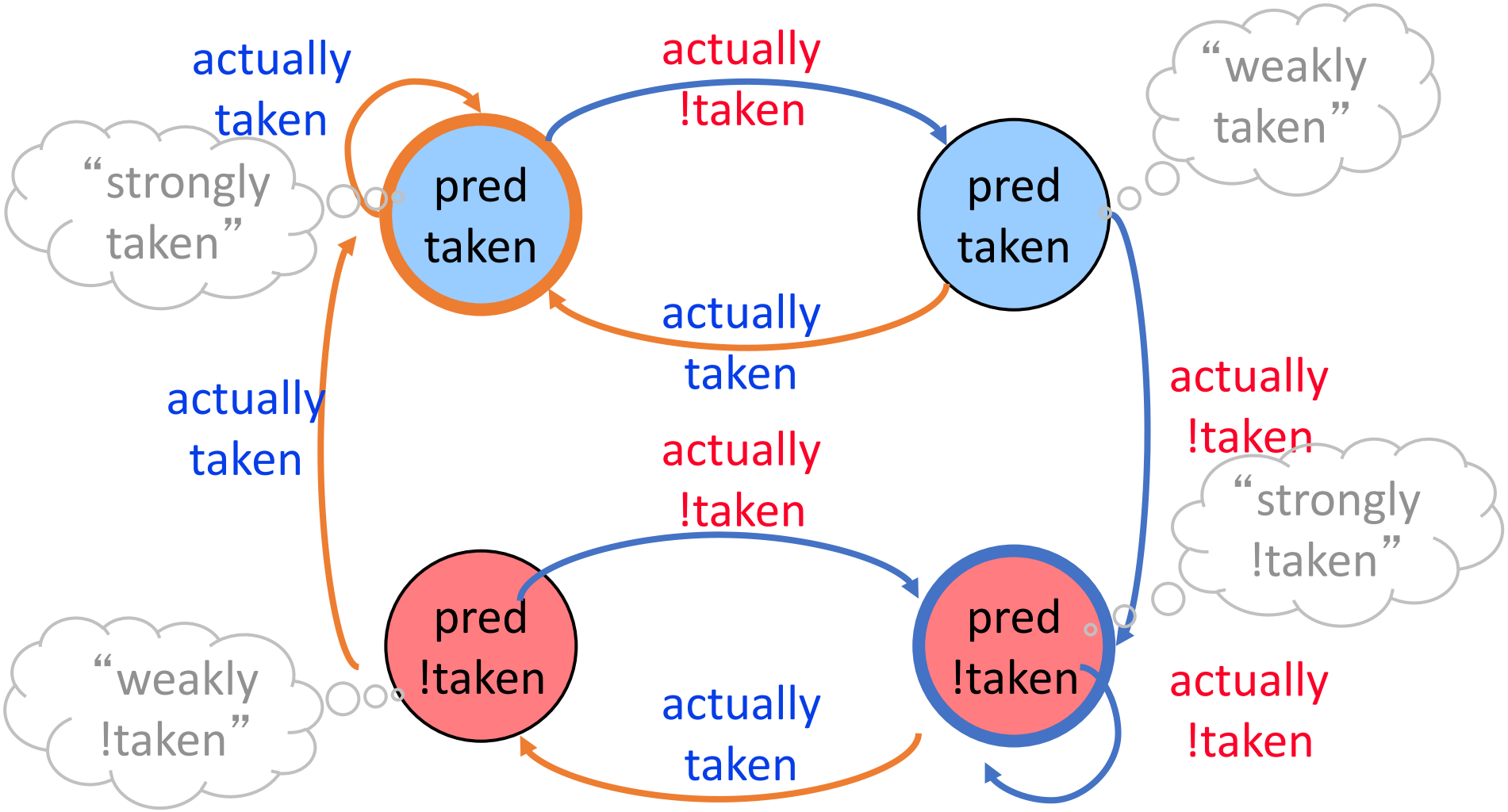
20% of all instructions are branches, 85% accuracy

Last-time predictor CPI =

[1 + (0.20*0.15) * 2] =

1.06

2-bit Predictors: A bit better



感謝