Microprocessor & Computer Architecture (µpCA)

UE19CS252

Dr. D. C. Kiran

Department of
Computer Science and Engineering

PES UNIVERSITY ONLINE

Microprocessor & Computer Architecture (µpCA)

Pipeline Processor: Branch Prediction

Dr. D. C. Kiran
Department of Computer Science and Engineering

Microprocessor & Computer Architecture (µpCA)

Syllabus

~~Unit 1: Basic Processor Architecture and Design~~

Unit 2: Pipelined Processor and Design

- ~~3-Stage ARM Processor~~
- ~~5-Stage Pipeline Processor~~
- ~~Introduction to Pipeline Processor~~
- ~~What May Go Wrong?~~
- ~~Introduction to Hazards, Stalls,~~
- ~~Structural Hazards~~
- ~~Data Hazard~~
  - ~~RAW, WAR, WAW Hazards~~
- ~~Attacking Data Hazard~~
  - ~~Software Approach~~
  - ~~Hardware Approach~~
- ~~Control Hazards~~
- Branch History Table
- Branch Prediction

Microprocessor & Computer Architecture (µpCA)

Text 1: "Computer Organization and Design", Patterson, Hennessey, 5th Edition, Morgan Kaufmann, 2014.

Reference 1:"Computer Architecture: A Quantitative Approach", Hennessey, Patterson, 5th Edition, Morgan Kaufmann, 2011.

Appendix C **Pipelining: Basic and Intermediate Concepts**

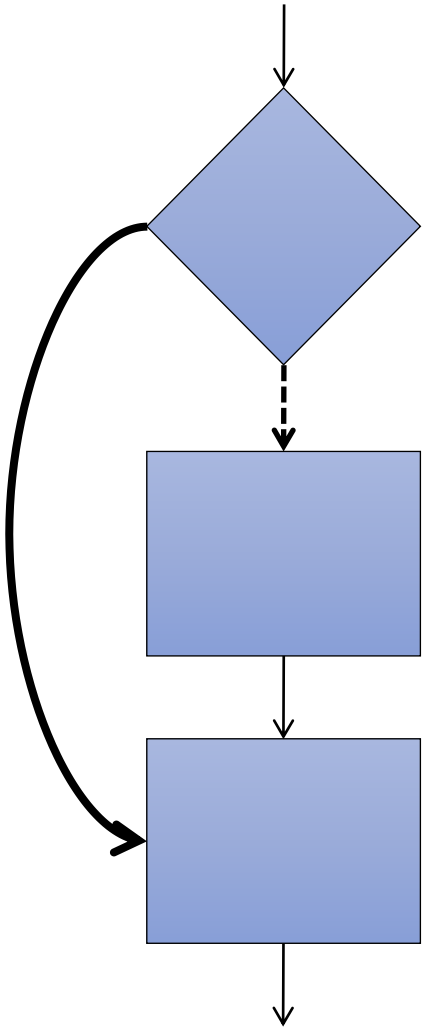
Branch Prediction

- Guessing the branch
  - One of the important problems in “Computer Architecture”
  - Static: prediction by compiler
  - Dynamic: prediction by hardware

- Static Prediction
  - Always Not Taken (easiest)
  - Always Taken
  - Taken / Not taken

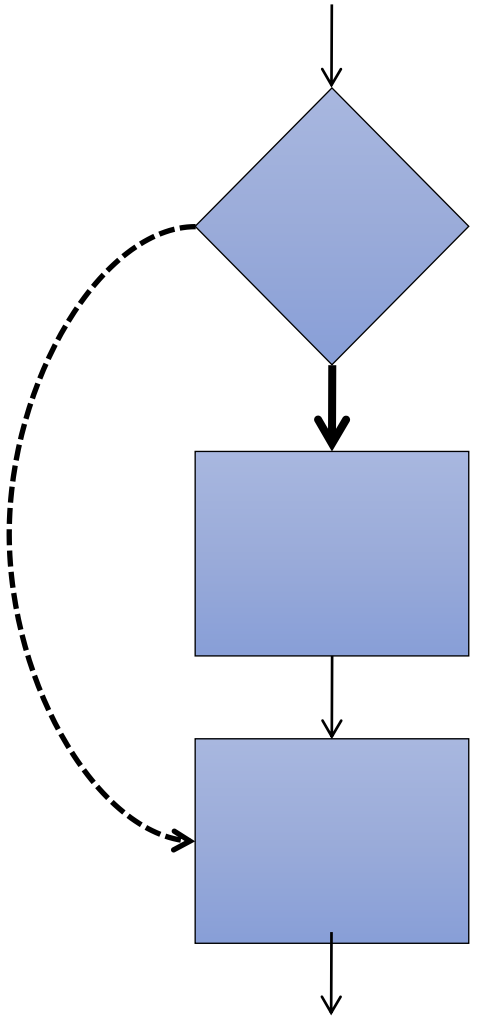
- Dynamic Prediction
  - 1- bit
  - 2-bit
  - others

Static Predictions



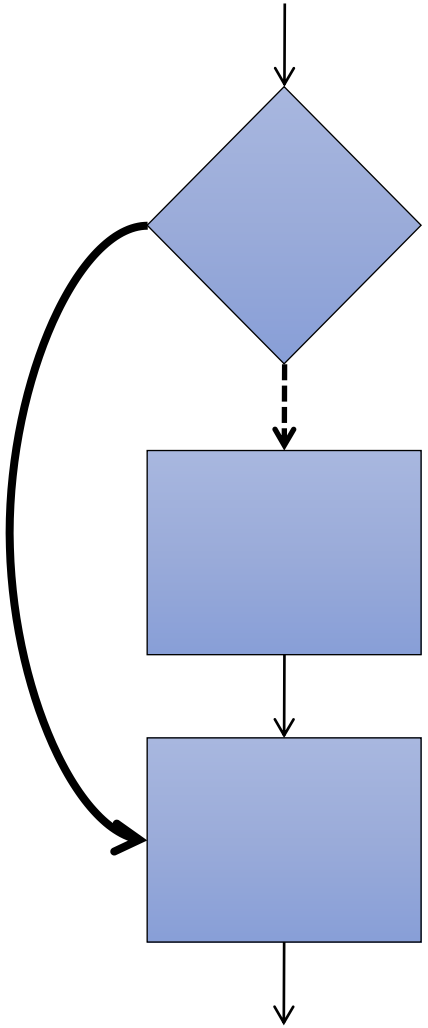
Always Taken

OR



Always Not Taken

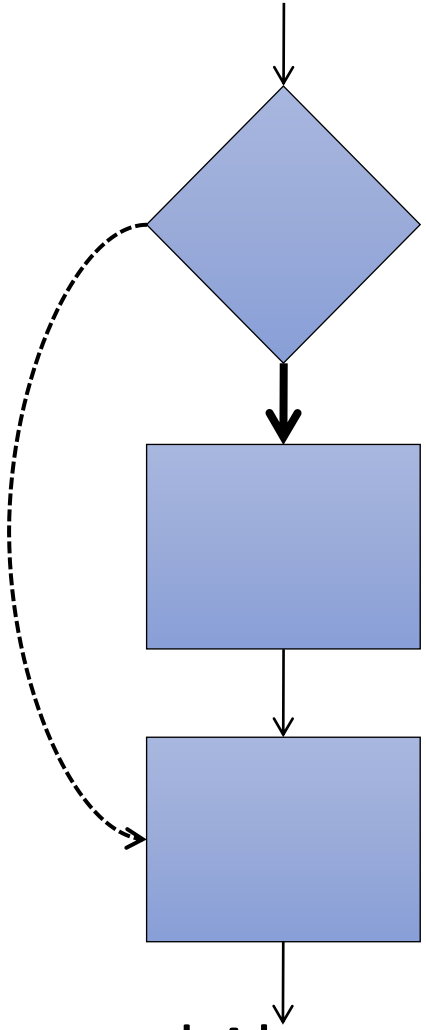
Static Predictions



Always Taken

- Early studies indicated that 2/3 of branches are taken
  - but 30% of those branches were unconditional!
- For conditional branches there appears to be no preferred direction.

Alternative Static Predictions

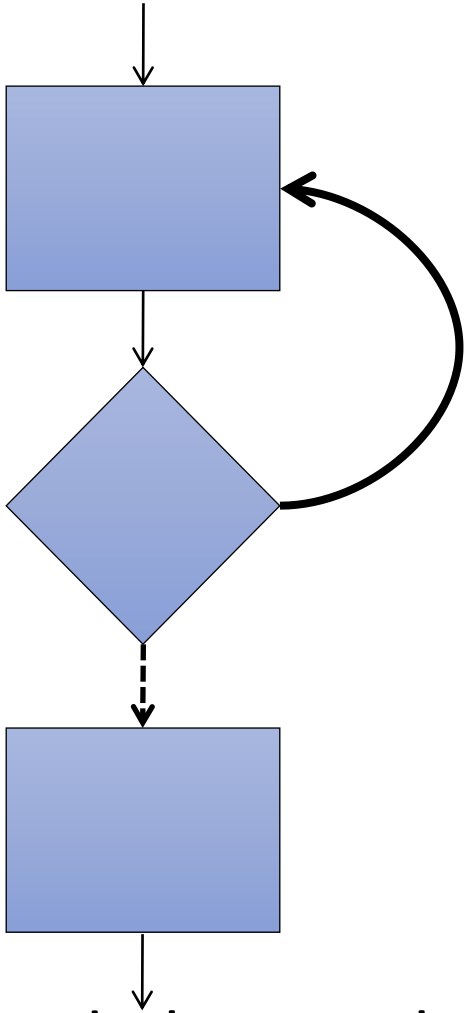


Forward Always Not Taken

Accuracy improvements are barely noticeable.

Static prediction based on profiling is slightly better.

Static branch-not-taken has no implementation cost on pipeline.



Backward Always Taken

Microprocessor & Computer Architecture (µpCA)

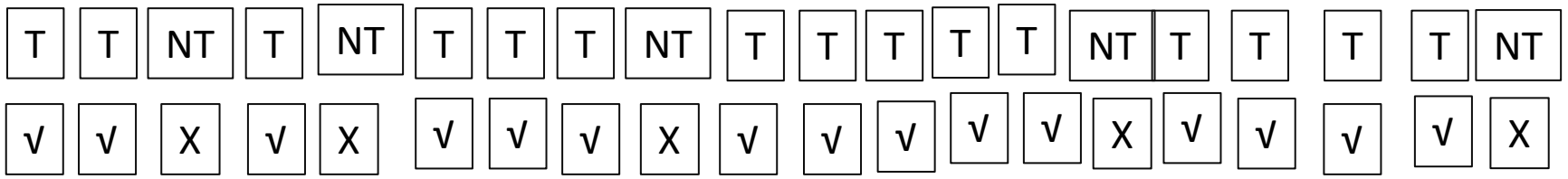
Static Prediction→ Taken

Consider a program with two branch statement with the following behavior

T T N T TN TT T T N T T T T TT N T T T T TNT

How many Miss Prediction ?

5 miss predictions



Microprocessor & Computer Architecture (µpCA)

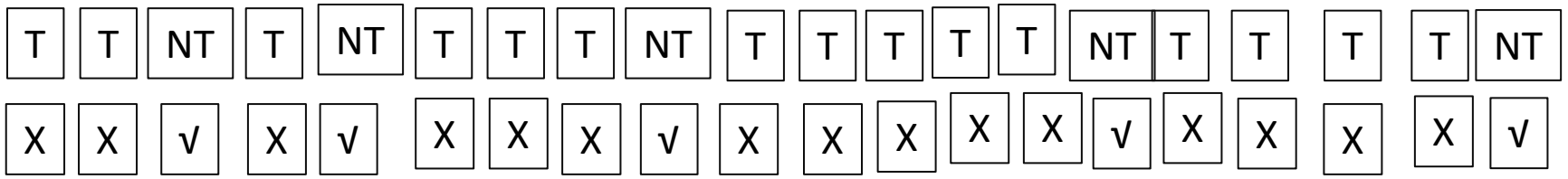
Static Prediction→ Not Taken

Consider a program with two branch statement with the following behavior

T T N T TN TT T T N T T T T TT N T T T T TNT

How many Miss Prediction?

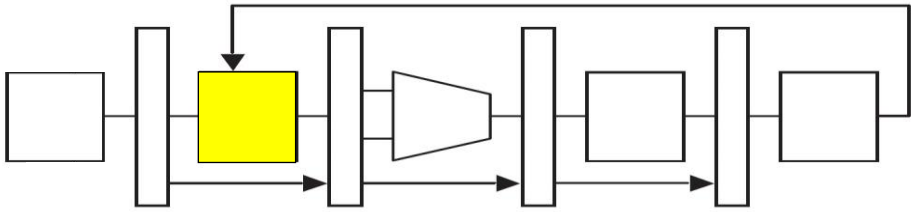
15 miss predictions



Dynamic Predictors

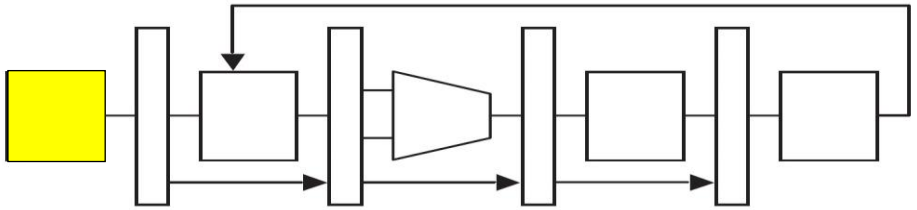
- Prediction of a given branch changes with the execution of the program.
  - **Simple**: a finite-state machine encodes the outcome of a few recent executions of the branch.
  - **Elaborate**: Not only early branch outcomes, but other correlated parts of the programs are considered.

When to Predict?



- Static prediction: at the Instruction Decode stage
  - Know that the instruction is a branch

- Dynamic prediction: at the Instruction Fetch stage
  - How to Know that the instruction is Branch
  - How to Know the target Address?
  - Should check Irrespective of branch instruction or not



How?

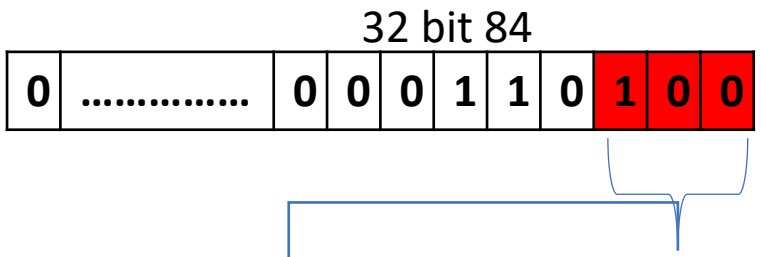
One-bit Predictor

- Simplest method:
  - A branch prediction buffer or Branch History Table (BHT) indexed by low address bits of the branch instruction.
  - Each buffer location (or BHT entry) contains one bit indicating whether the branch was recently taken or not.
  - Change bit on misprediction
  - Always mispredicts in first and last loop iterations.

Microprocessor & Computer Architecture (µpCA)

Branch History Table or Branch Prediction Buffer:-> One Bit

Address	Branch Address	Target Address	Prediction
000	432	456	1
001	97	123	0
010	130	143	1
011	67	98	0
100	84	244	1
101	261	532	1
110	518	786	1
111	1031	1134	0

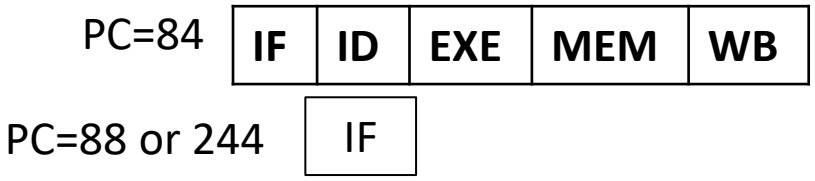


Ex:

000080 : add R1, R2, R3
0000084: beq R1, R2, 244
0000088: sub R1, R4, R5
orr R2, R7, R8

....

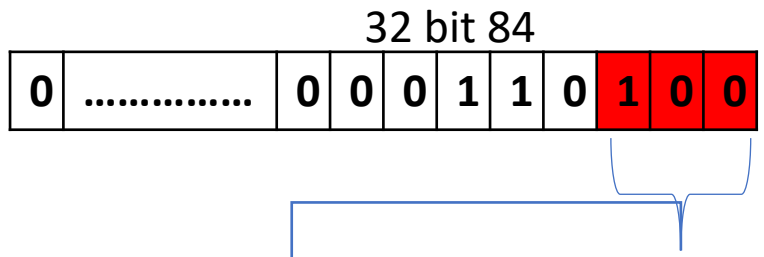
0000244: add R1, R4, R5



Microprocessor & Computer Architecture (µpCA)

Branch History Table or Branch Prediction Buffer:-> One Bit

Address	Branch Address	Target Address	Prediction
000	432	456	1
001	97	123	0
010	130	143	1
011	67	98	0
100	84	244	1
101	261	532	1
110	518	786	1
111	1031	1134	0



PC=244

IF	ID	EXE	MEM	WB

IF	ID	EXE	MEM	WB

If Branch Taken, update PC

Ex:

000080 : add R1, R2, R3
0000084: beq R1, R2, 244
0000088: sub R1, R4, R5
orr R2, R7, R8

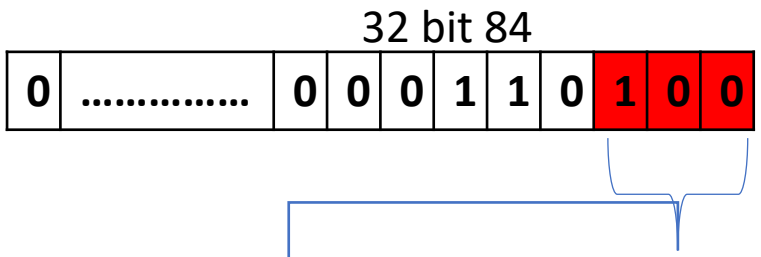
....

0000244: add R1, R4, R5

Microprocessor & Computer Architecture (µpCA)

Branch History Table or Branch Prediction Buffer:-> One Bit

Address	Branch Address	Target Address	Prediction
000	432	456	1
001	97	123	0
010	130	143	1
011	67	98	0
100	84	244	**0**
101	261	532	1
110	518	786	1
111	1031	1134	0



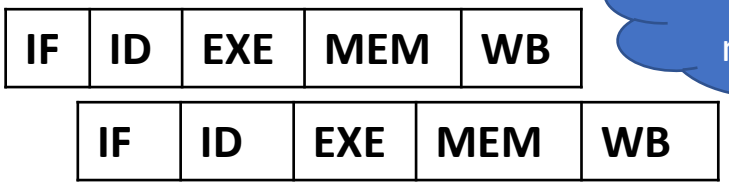
Ex:

000080 : add R1, R2, R3
0000084: beq R1, R2, 244
0000088: sub R1, R4, R5
orr R2, R7, R8

....

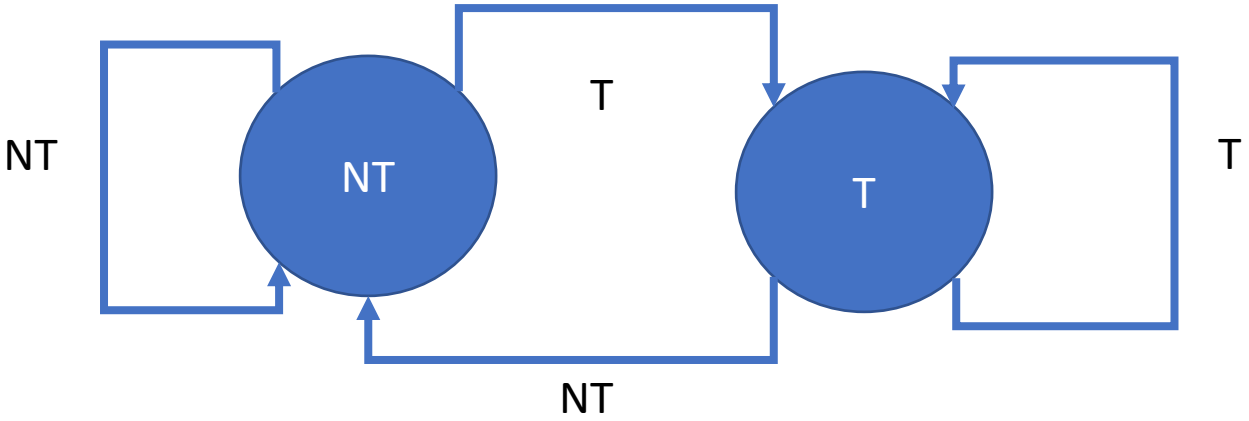
0000244: add R1, R4, R5

PC=88

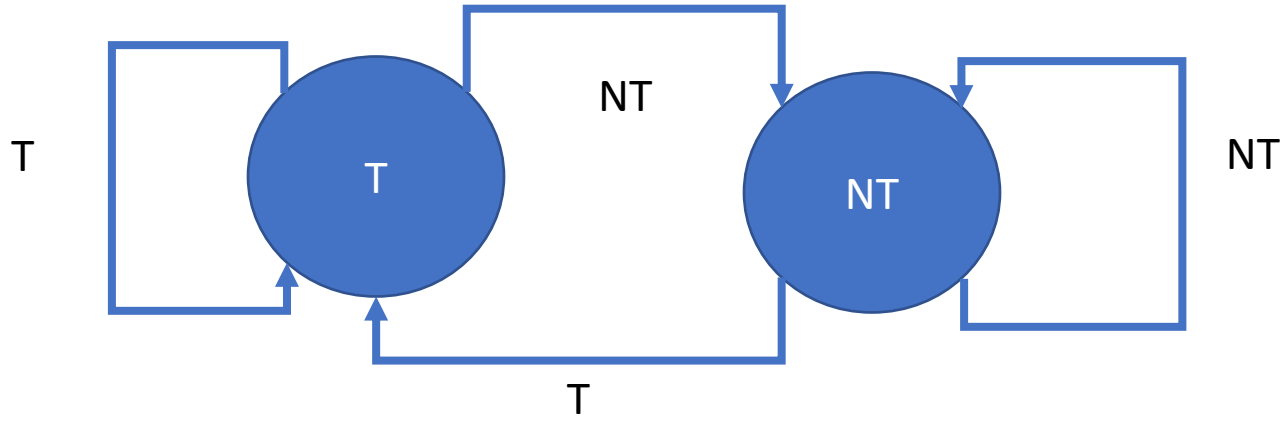


If Branch Not Taken
Continue with
normal execution

1 Bit Branch Prediction

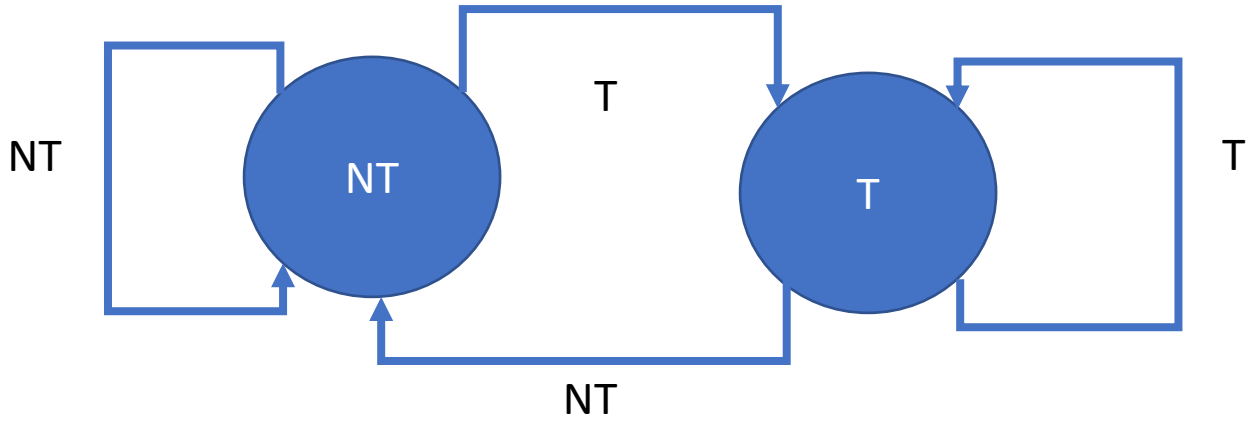


OR



Microprocessor & Computer Architecture (µpCA)

1 Bit Branch Prediction



Consider a program with two branch statement with the following behavior

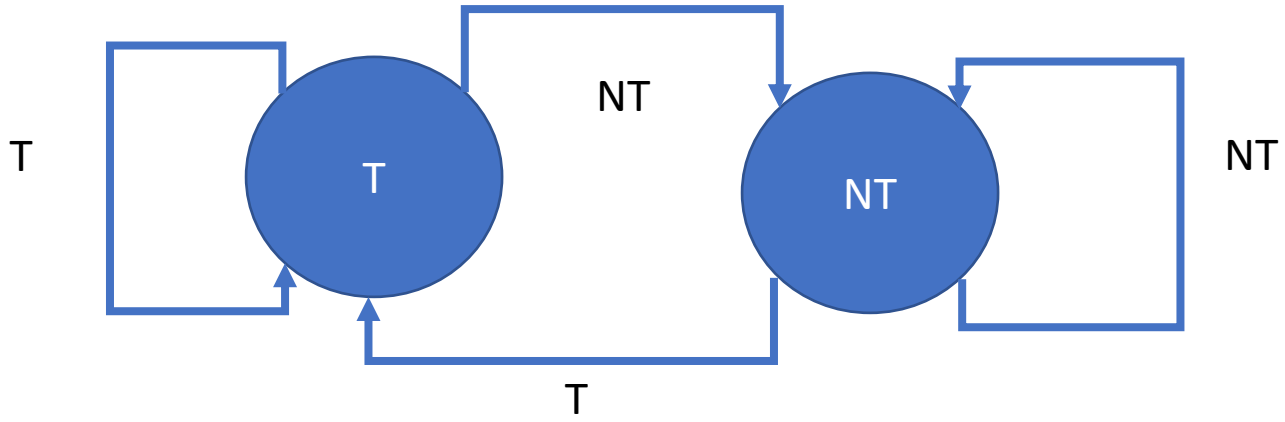
T T N T TN TT T T N T T T T TT N T T T T TNT

How many Miss Prediction if the Initial state is NT? **10 miss predictions**

T	T	NT	T	NT	T	T	T	NT	T	T	T	T	T	NT	T	T	T	T	NT
X	√	X	X	X	X	√	√	X	X	√	√	√	√	X	X	√	√	√	X

Microprocessor & Computer Architecture (µpCA)

1 Bit Branch Prediction



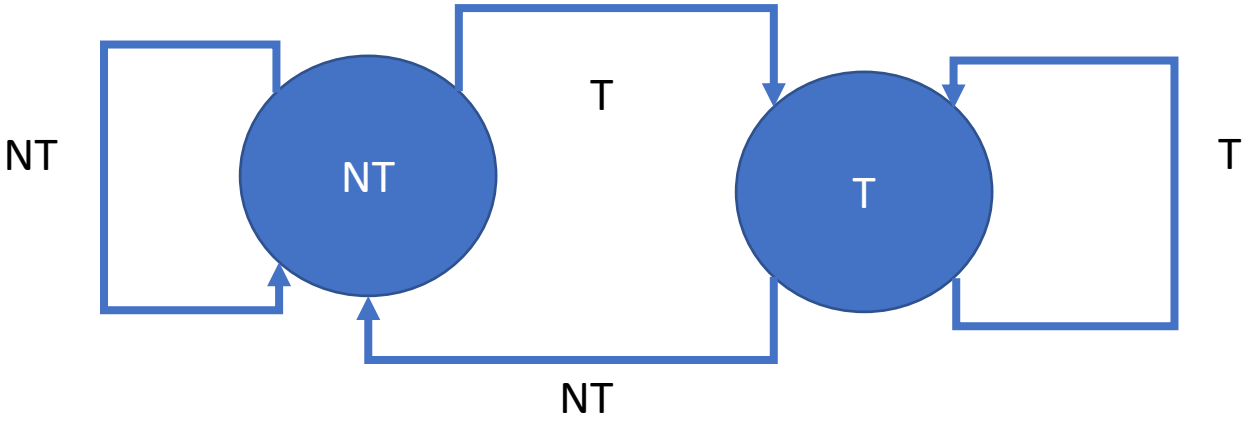
Consider a program with two branch statement with the following behavior

T T N T TN TT T T N T T T T TT N T T T T TNT

How many Miss Prediction if the Initial state is NT? **9 miss predictions**

T	T	NT	T	NT	T	T	T	NT	T	T	T	T	T	NT	T	T	T	T	NT
√	√	X	X	X	X	√	√	X	X	√	√	√	√	X	X	√	√	√	X

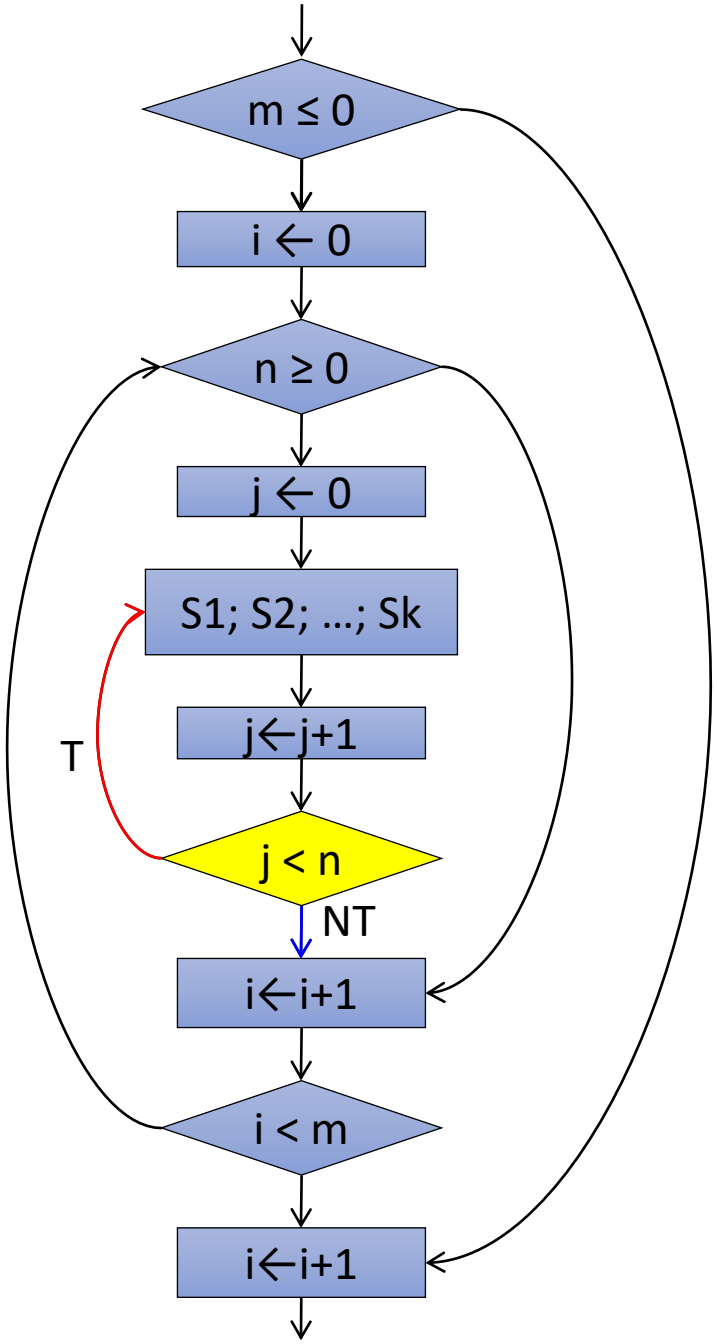
1 Bit Branch Prediction




```
for(j=0; j<n ; j++)
        begin S1; S2; …; Sk end;
```

	1-bit	
i	**Pred**	**Outc**
0	NT	T
1	T	T
n	T	NT
0	NT	T
1	T	T

2 Misprediction for each loop




```
for(i=0 ; i < m ; i++)
    for(j=0; j<n ; j++)
        begin S1; S2; …; Sk end;
```

		1-bit	
i	**j**	**Pred**	**Outc**
0	0	NT	T
0	1	T	T
0	n	T	NT
1	0	NT	T
1	1	T	T

2 × *m* mispredictions

2 Bit Branch Prediction

THANK YOU

Dr. D. C. Kiran
Department of Computer Science and Engineering

dckiran@pes.edu

9829935135