CE/CZ 3001:
Advanced Computer Architecture

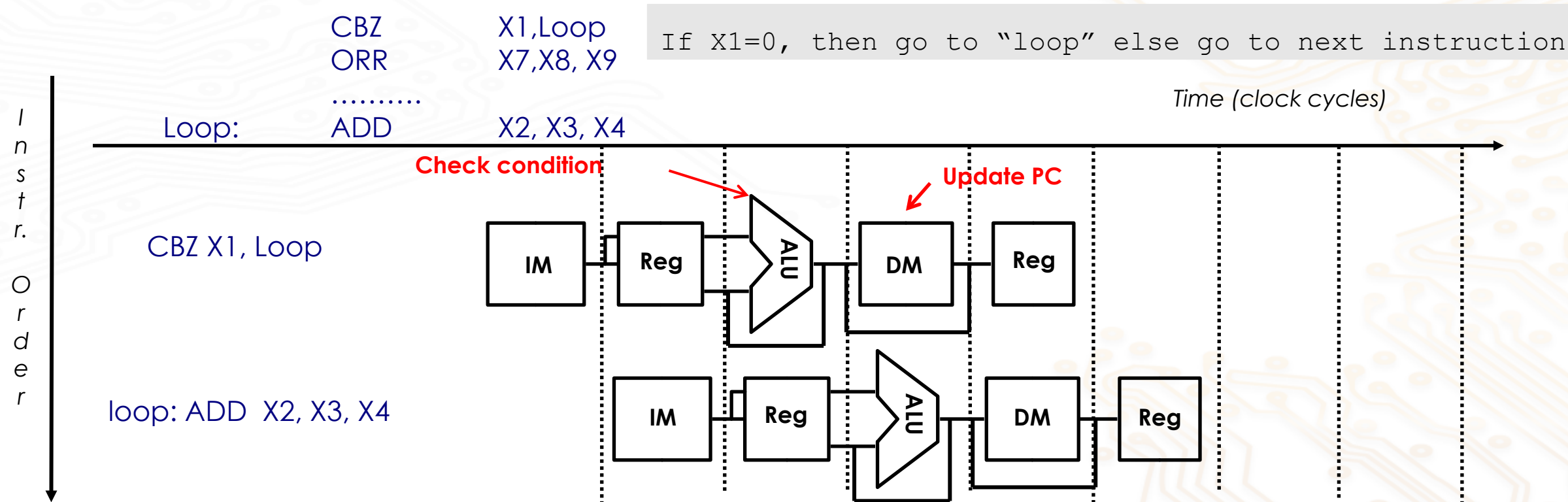**(Module 4: Instruction Level Parallelism(ILP))**

Dr Smitha K. G.
School of Computer Science
And Engineering

Summary of pre video

- Control Hazards

- How to tackle control hazards

- Conservative way

- Early evaluation of the content of PC (using hardware change)

- Branch prediction
  - Static branch prediction
  - Dynamic branch prediction

Control Hazards

- Branching instructions
  - pipeline to stall as it changes the execution sequence of instructions.

CBZ   X1,Loop
ORR   X7,X8, X9

If X1=0, then go to "loop" else go to next instruction

....

Loop: ADD   X2, X3, X4

Time (clock cycles)

Check condition

Update PC

Instr. Order

CBZ X1, Loop

IM | Reg | ALU | DM | Reg

loop: ADD X2, X3, X4

IM | Reg | ALU | DM | Reg

3

How to tackle control hazards

- **Conservative way**

- **Branch prediction**

Conservative method

Example

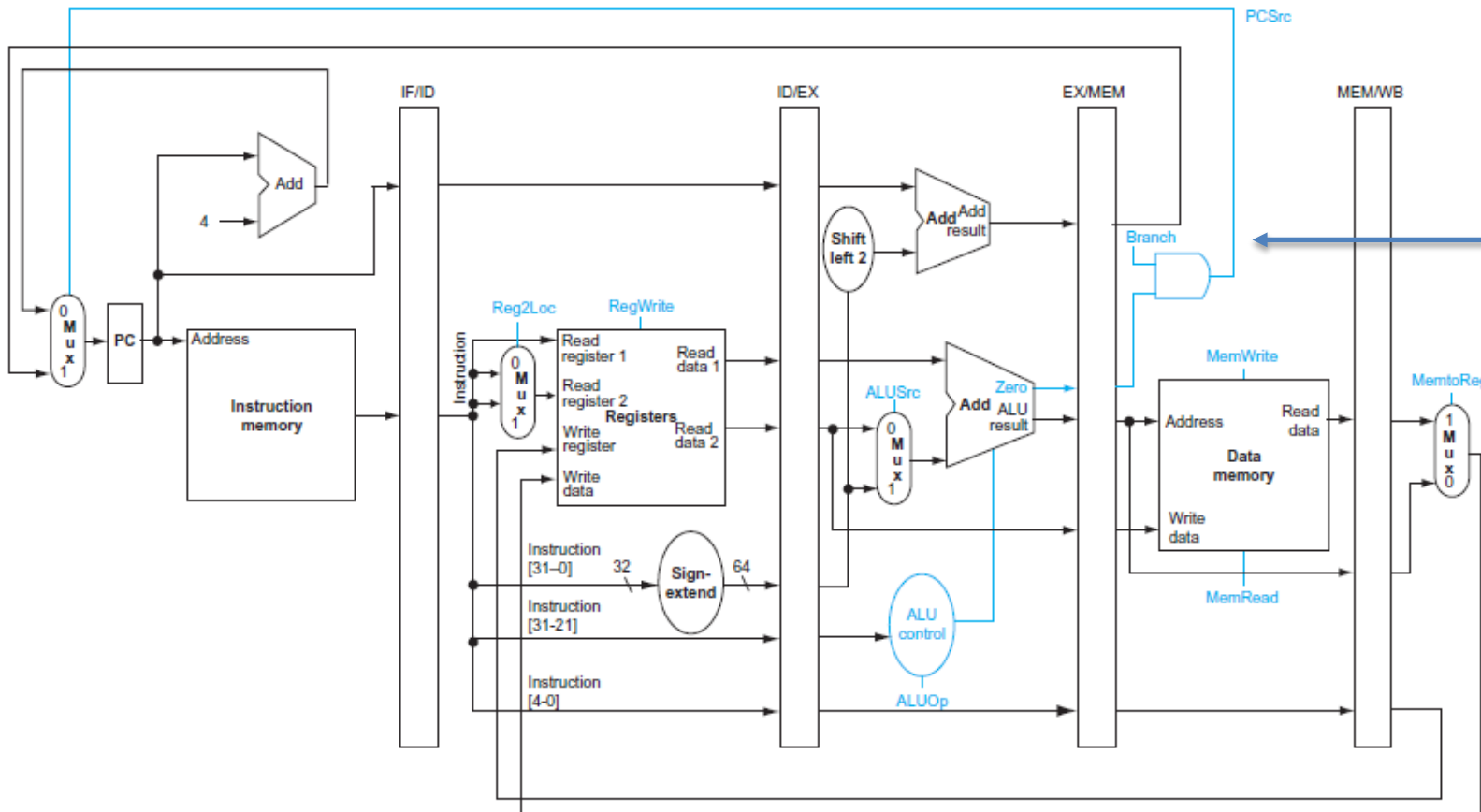   CBZ X1, loop
        ADD X2, X3, X4
                ....
Loop: LDUR X3, [X0, #30]

Clock	1	2	3	4	5	6	7	8	9
I1	IF	ID	EX	M	WB				
I2		nop	nop	nop	nop	nop			
I3			nop	nop	nop	nop	nop		
I4				nop	nop	nop	nop	nop	
I5						IF	ID	EX	M	WB

CBZ updating PC in MEM stage Branch penalty= 3 stalls

If PCSrc and Branch Target Address is updated in execute stage (hardware changes):
Then CBZ can update PC in EXE stage. Branch penalty= 2 stalls

5

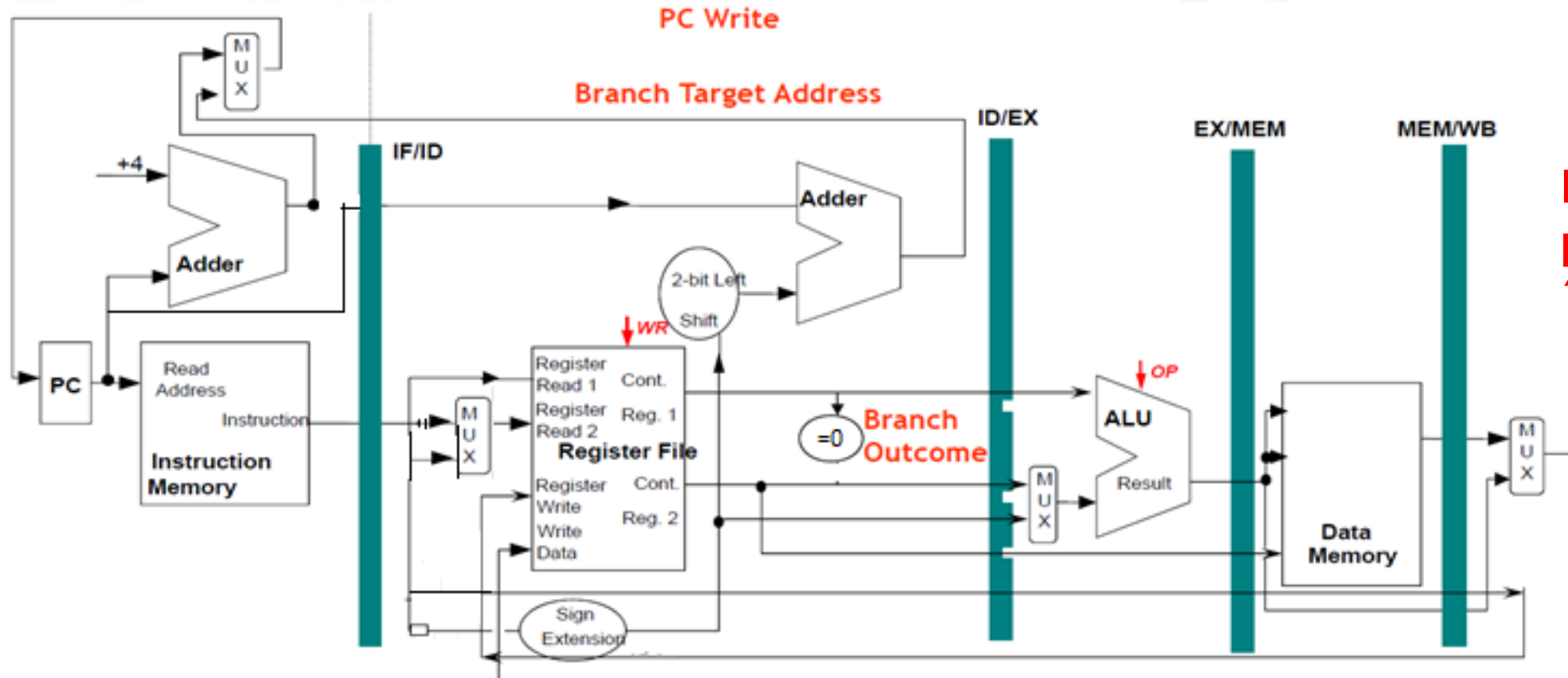# Early Evaluation of PC for reducing branch stalls

Example

CBZ X1, loop
ADD X2, X3, X4
....
Loop: LDUR X3, [X0, #30]

Clock	1	2	3	4	5	6	7	8	9	10	11
I1	IF	ID	EX	M	WB						
I2		nop	nop	nop	nop	nop					
I5			IF	ID	EX	M	WB				

PC Write

Branch Target Address

Branch penalty= 1 stalls

Example

1 Loop:	**LDUR**	**X0, [X1, #0]**
2	**LDUR**	**X6, [X1, #-8]**
3	**LDUR**	**X10, [X1,#-16]**
4	**LDUR**	**X14, [X1,#-24]**
5	**ADD**	**X4, X0, X2**
6	**ADD**	**X8, X6, X2**
7	**ADD**	**X12, X10, X2**
8	**ADD**	**X16, X14, X2**
9	**STUR**	**X4, [X1, 0]**
10	**STUR**	**X8, [X1, #-8]**
11	**STUR**	**X12, [X1, #-16]**
12	**STUR**	**X16, [X1, #-24]**
13	**SUBI**	**X1, X1, #32**
14	**CBNZ**	**X1,LOOP**

Loop unrolled result (in slide 40)

Removed all data hazards **(full data forwarding)**

But there is a **control hazard**

CPI=

Another example- from exam paper

```
I1              ADDI X3, X31, 0X24

I2  Loop:       LDUR X2, [X1, #0]

I3               ADDI X2, X2, #10

I4               STUR X2, [X1, #0]

I5               ADDI X1, X1, #8

I6               SUBI X3, X3, #1

I7               CBNZ X3    Loop
```

find steady state CPI – no data forwarding (WB and dec simultaneously, Brach target address is updated in decode stage)

Branch Prediction

Branch prediction

Static branch prediction techniques:- The actions for the branch are fixed for each branch during the entire execution. (when behaviour is highly predictable).

Dynamic branch prediction techniques:- The prediction decision may change depending on the execution history (when behaviour is not predictable).

Static Prediction

a) **Branch Always Not Taken – Predict-Not-Taken (Speculation)**

- Execute successive instructions in sequence.

- Flush the pipeline and read correct instructions if branch actually taken

i1:	ADD	X3, X1, #2
i2:	CBZ	X3, L1
i3:	ADD	X1, X0, X0
i4:	AND	X4, X1, X2
i5: L1	SW	X4,[X5,#0]

(Assume branch solved in ID stage)

Instr.	1	2	3	4	5	6	7	8
I1	IF	ID	EX	M	WB			
I2		IF	ID	--	--	--		
I3			IF	F	F	F	F	
I5				IF	ID	EX	M	WB

Branch taken

1. Need to **flush** the next instruction already fetched.

2. Restart the execution by fetching the instruction at the branch target address – **One-cycle penalty.**

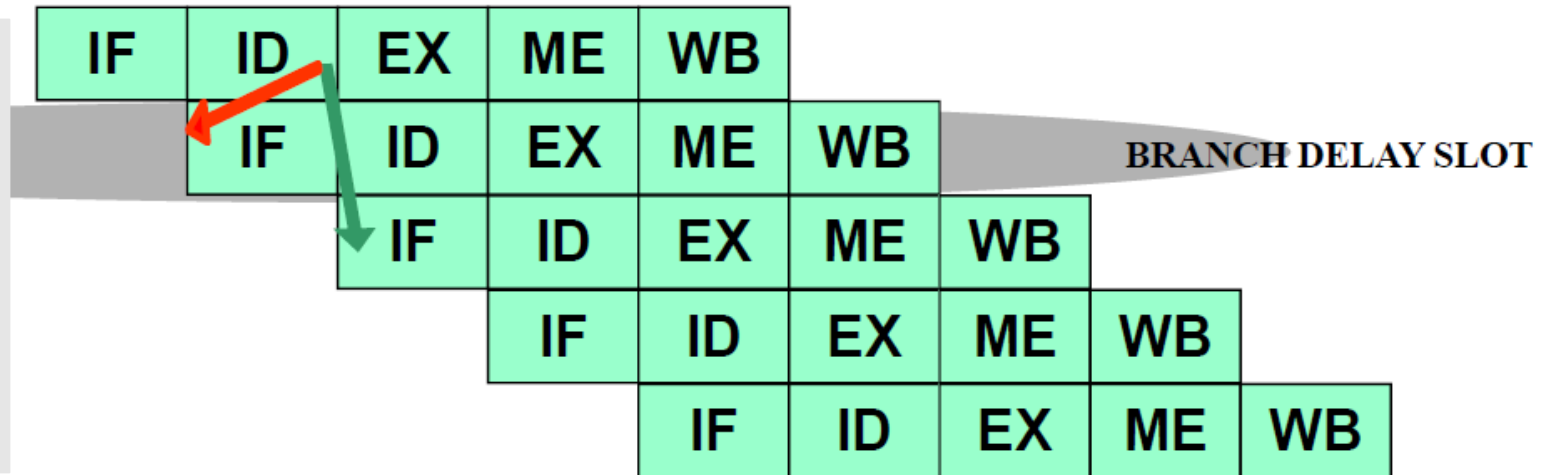| Instr.	1	2	3	4	5	6	7	8
I1	IF	ID	EX	M	WB			
I2		IF	ID	--	--	--		
I3			IF	ID	EX	M	WB	
I4				IF	ID	EX	M	WB
I5					IF	ID	EX	M

Branch not taken

Static Prediction – Delayed Branch

- The compiler statically schedules an independent instruction in the **branch delay slot**.

```
CBZ     X1 L1
ADD     X4,X5,X6
L1  LDUR    X3, [X0, #300]
    LDUR    X7, [X0, #400]
    LDUR    X8, [X0, #500]
```
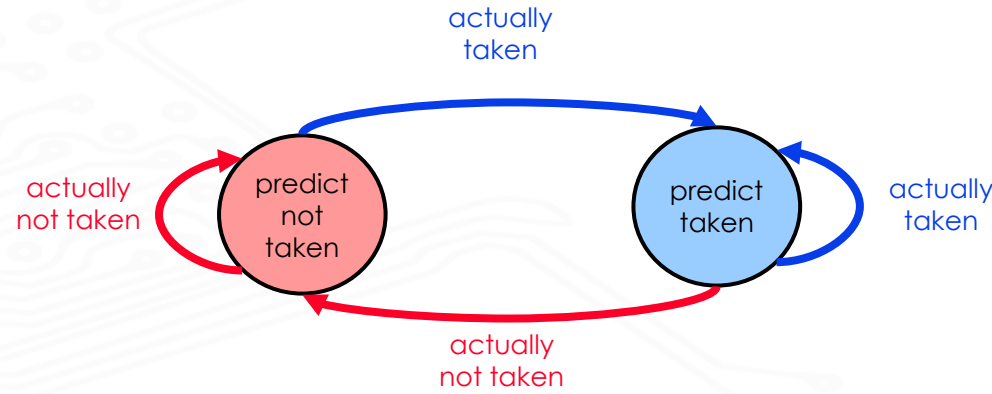

A previous **add** instruction with no effects on the branch is scheduled in the **Branch Delay Slot.**

Dynamic Prediction- Extra hardware required

Scheme 1: Single T bit (Last time predictor)

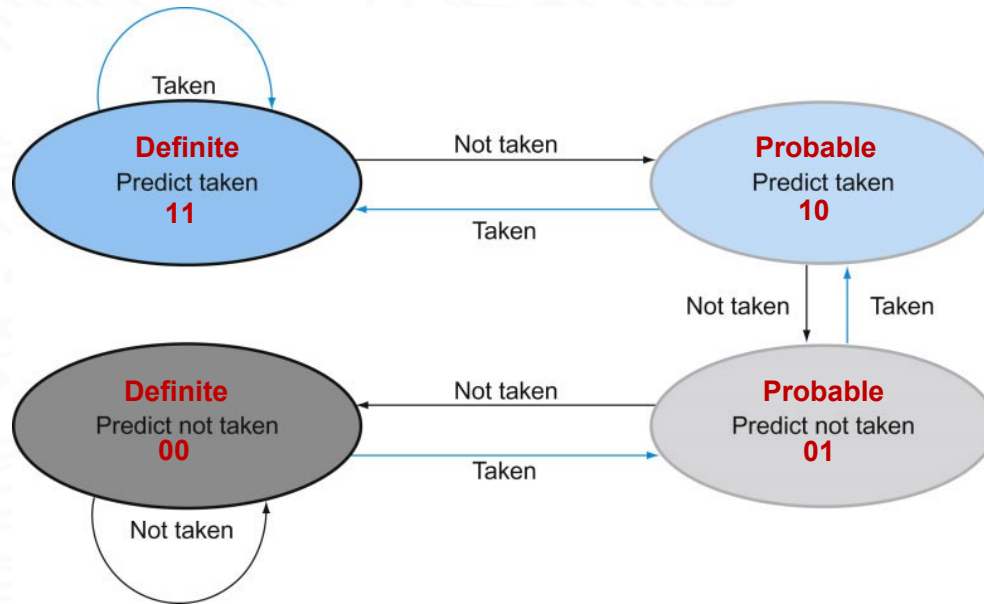- T is set to 1 when a branch is confirmed taken , 0 when not.

Example:
Assume single bit predictor for branch b1. Initial value of predictor to be 0. Then T_{b1} predicts that the branch is not taken.

branch b1 taken, T_{b1} =1 (predict next branch taken)
branch b1 not taken, T_{b1} =0 (predict next branch not taken)

Dynamic Prediction (two bit prediction)

Scheme 2: 2 bit prediction uses the result of last two branches to predict instead of just the last branch.

- TNTNTNTNTNTNTNTNTN → 50% accuracy

- (assuming initial to probable (weakly taken)

- Disadvantage: More hardware

3. (a) Consider the following repeating sequence of actual outcome for a branch (N N T N T N). Here 'N' means that the branch is not taken and 'T' means that branch is taken. Assume that there is only one branch instruction in the program. The predictors are initialized to 'weakly taken' stage.
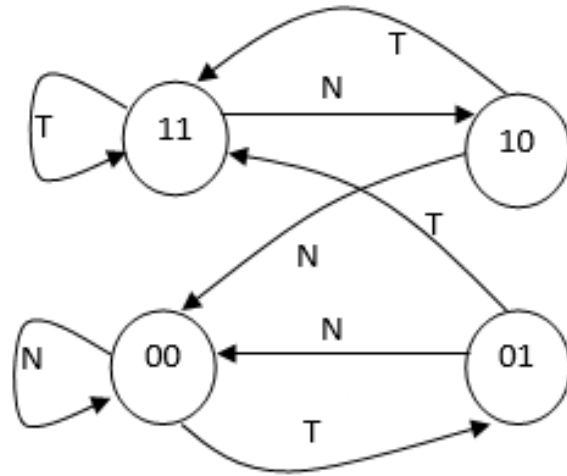
11 → Strongly taken

10 → Weakly taken

01 → Weakly not taken

00 → Strongly not taken

ii. Compare the accuracy with an always not taken static predictor and comment on the best choice of prediction for the above case.

Figure Q3a

(4 marks)

i. Find the prediction accuracy of two-bit predictor shown in Figure Q3a by properly indicating the prediction decision at each stage.

(7 marks)

state	10										
prediction											
Actual											

14

Dynamic Prediction (Part 2/2)

- **Scheme 3: Bimodal prediction** uses a counter and the state of that counter determines the prediction.

- Generalized scheme of 3 bit predictor.

- The counter is incremented if branch is taken.

- The counter is decremented if branch not taken.

- Counters saturate (no wraparound). The speculation decision is based on the most significant bit: if MSB is 1, then counter is above half way.

000 → 001 → 010 → 011 → 100 → 101 → 110 → 111