



CS230: Digital Logic Design and Computer Architecture

Lecture 14: Branch Prediction and Interrupts

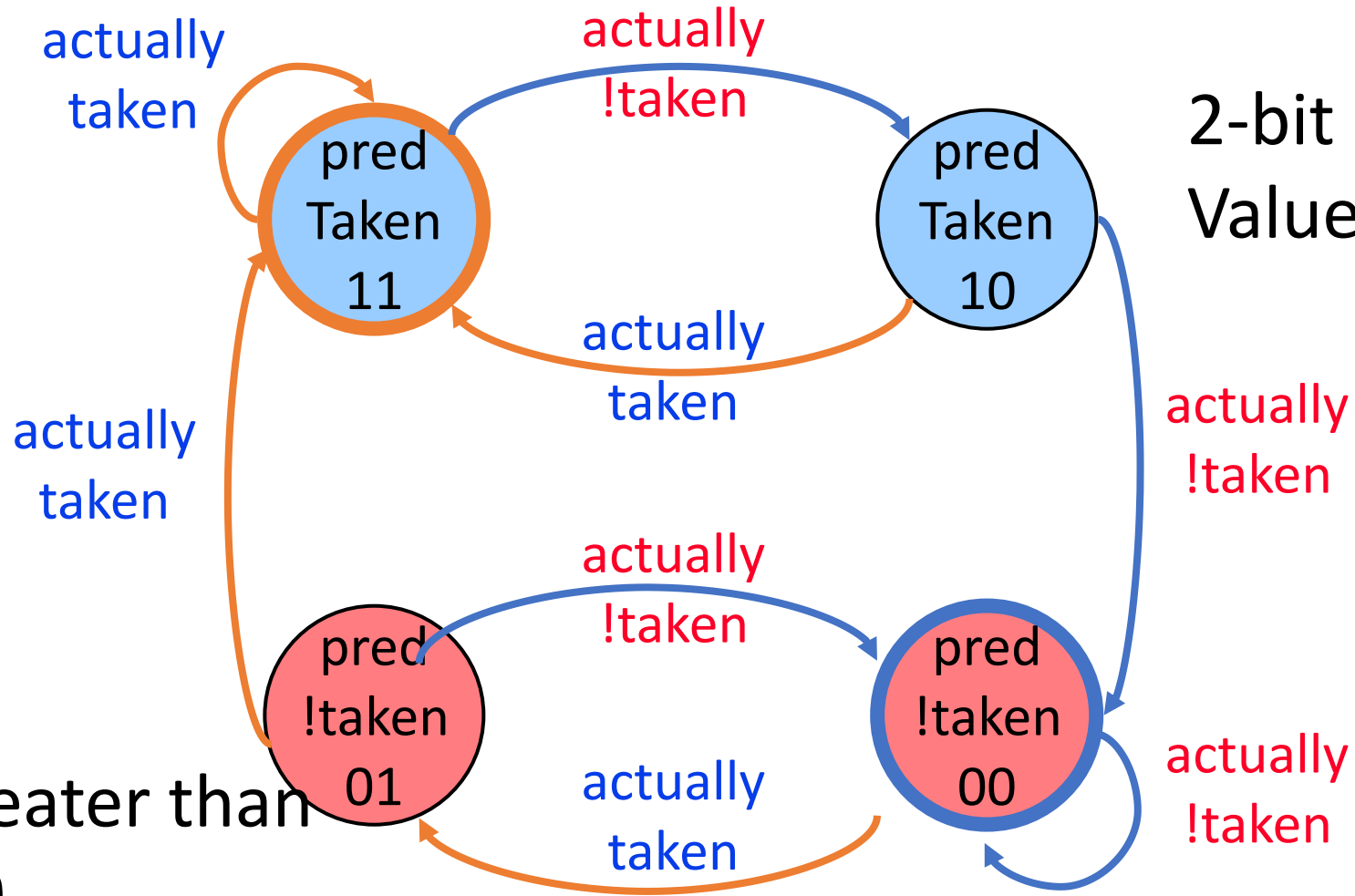
<https://www.cse.iitb.ac.in/~biswa/courses/CS230/main.html>

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

Phones
(smart/non-smart)
on silence plz,
Thanks



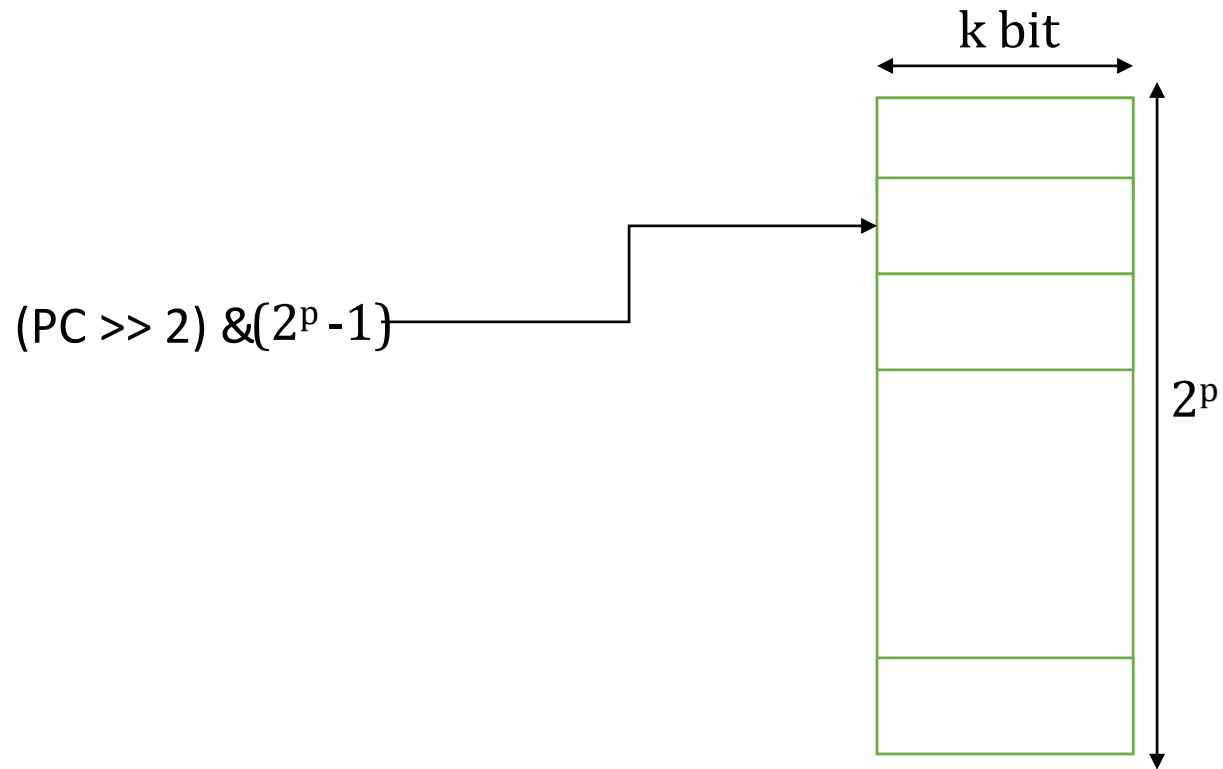
2-bit Bimodal Predictors: A bit better



2-bit saturating counter:
Values from 00 to 11

Counter greater than
equal to 10,
prediction: taken

No history predictor: 2 bit predictor



Bimodal predictor: Good for biased branches

Local and global history

- **Local Behavior**

What is the predicted direction of Branch A given the outcomes of previous instances of Branch A ?

- **Global Behavior**

What is the predicted direction of Branch Z given the outcomes of *all** previous branches A, B, ..., X and Y?

Number of previous branches tracked limited by the history length

Two Level Branch Predictors

First level: **Global branch history register** (N bits)

The direction of last N branches

Second level: **Table of saturating counters for each history entry**

The direction the branch took the last time the same history was seen



GHR
(global history register)

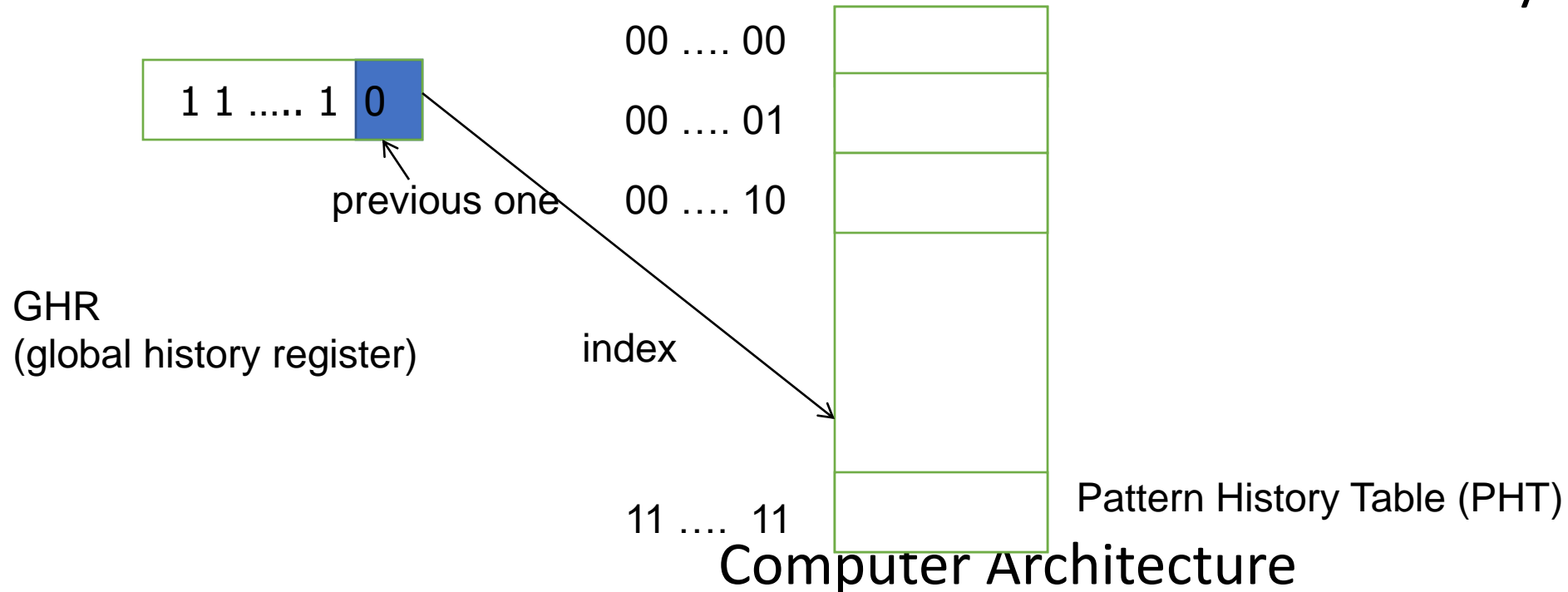
Two Level Branch Predictors

First level: **Global branch history register** (N bits)

The direction of last N branches

Second level: **Table of saturating counters** for each history entry

The direction the branch took the last time the same history was seen



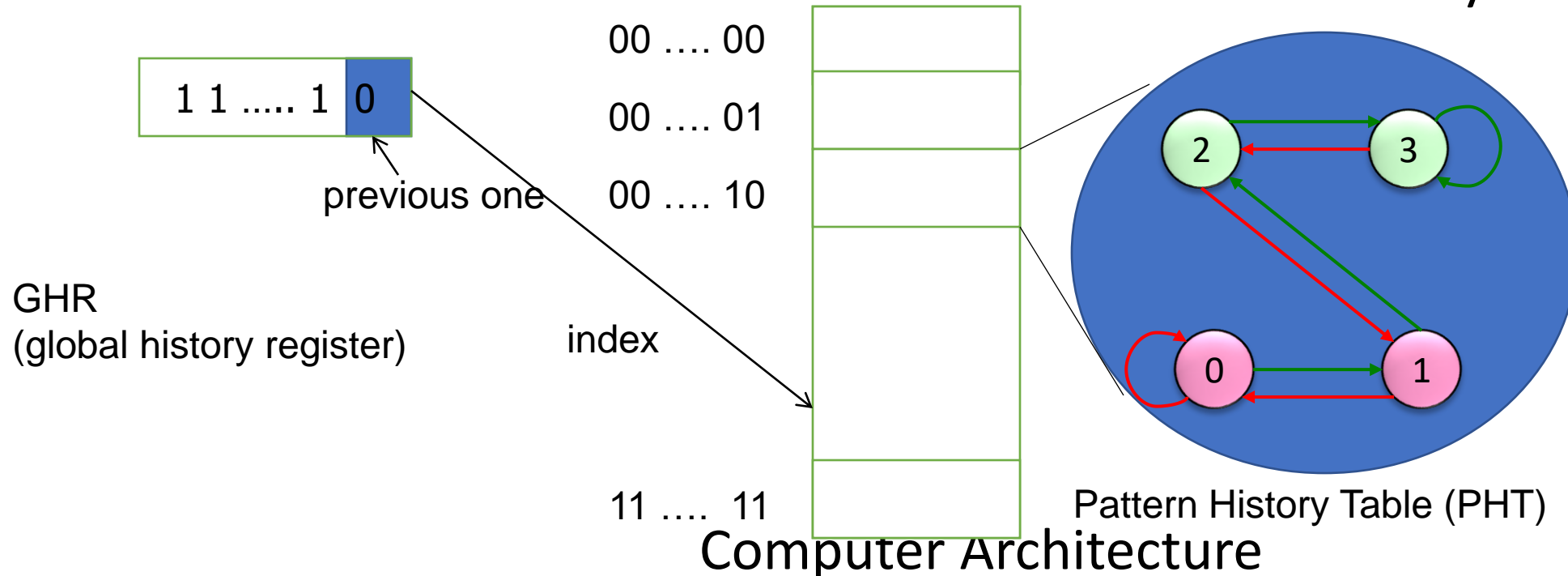
Two Level Branch Predictors

First level: **Global branch history register** (N bits)

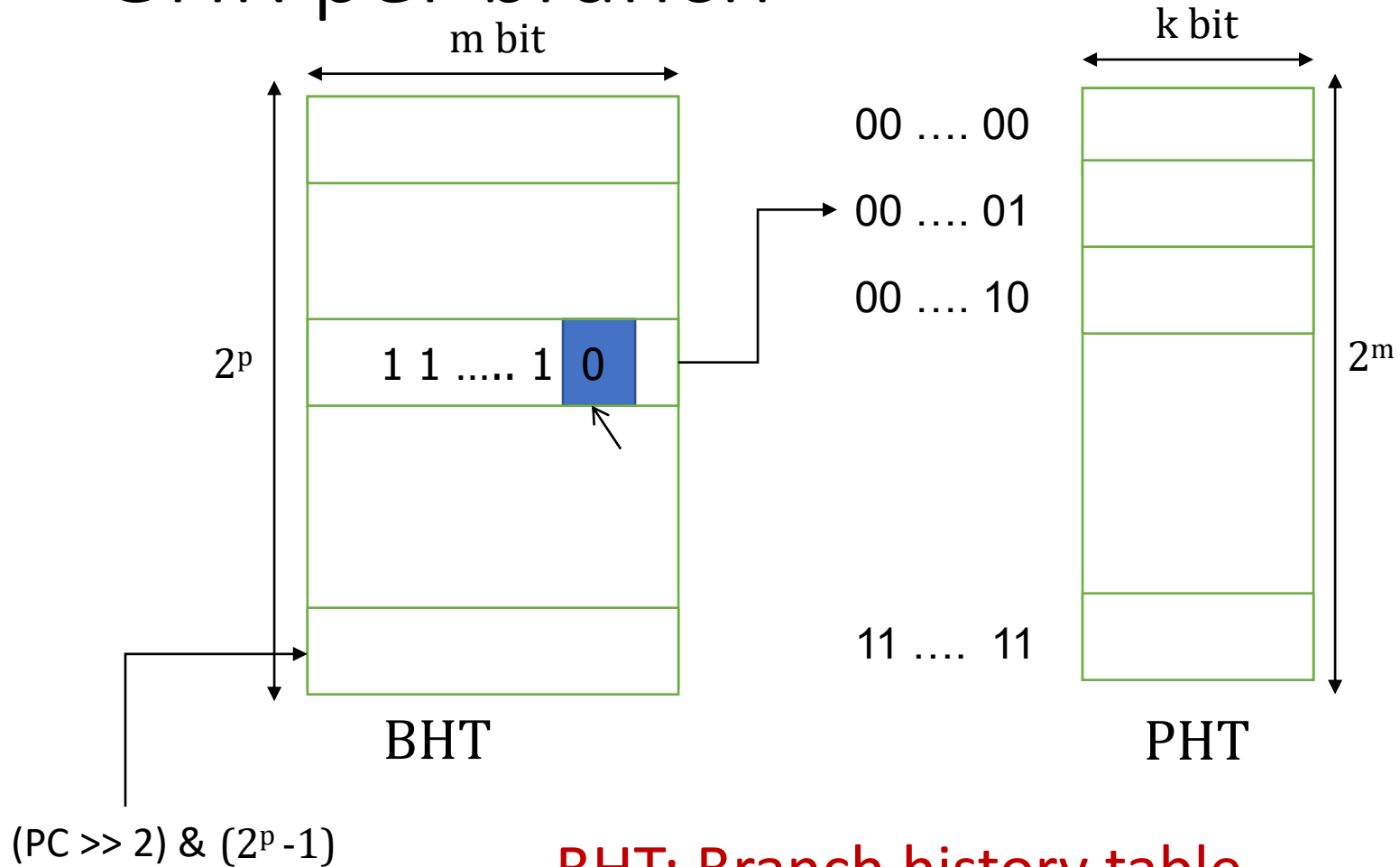
The direction of last N branches

Second level: **Table of saturating counters for each history entry**

The direction the branch took the last time the same history was seen



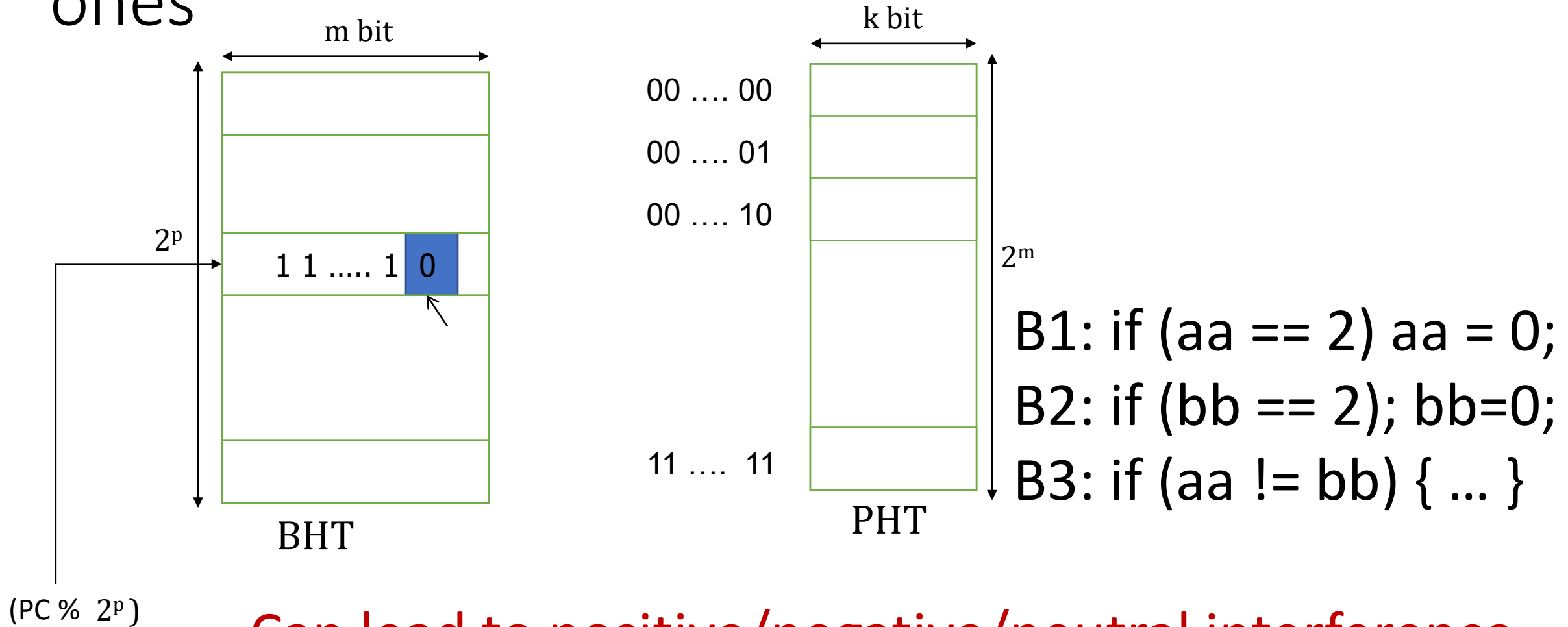
GHR per branch



BHT: Branch history table

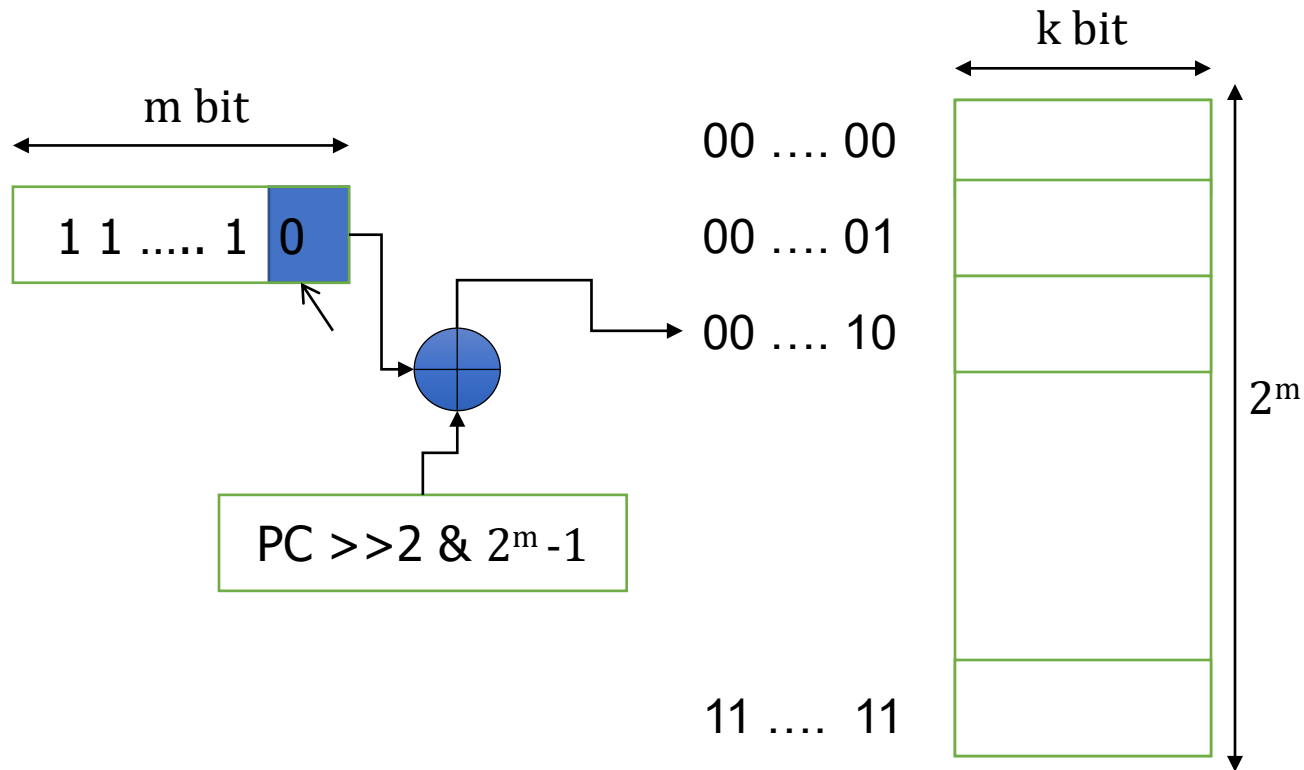
Mostly $K=2, m=12$ for example
Computer Architecture

Set of branches: One register for correlated ones



Can lead to positive/negative/neutral interference

Gshare is the answer



For a given history and for a given branch (PC) counters are trained

Few Important Points

Branch prediction happens at the IF stage.

We know the target outcome at the end of EX stage.

So BHT and PHT will be updated after EX stage for the corresponding PC. Any issues here?

Issue

```
I1  F  D  E
I2   F  D  E
I3   F  D  E
```

Lets assume I1 and I3 are branch instructions. I1 will update BHT and PHT in E stage, and I3 will probe BHT and PHT in F stage. To make sure PHT is updated correctly with the correct BHT entry, BHT entry is communicated till the E stage.

State-of-the-art

State of the art: Neural vs. TAGE

1970: Flynn

1972: Riseman/Foster

1979: Smith Predictor

1991: Two-level prediction

1993: gshare, tournament

1996: Confidence estimation

1996: Vary history length

1998: Cache exceptions

2001: Neural predictor

2004: PPM

2006: TAGE

2016: Still TAGE vs Neural

- Neural: AMD, Samsung
- TAGE: Intel?, ARM?

Similarity

– Many sources or “features”

- Key difference: how to combine them
 - TAGE: Override via partial match
 - Neural: integrate + threshold
- Every CBP is a cage match
 - Andre Seznec vs. Daniel Jimenez



BTB (Target Address Predictor)

Address of branch instruction

0b0110 [...] 01001000

Branch instruction

BNEZ R1 Loop

Branch Target Buffer (BTB)

30-bit address tag

target address

0b0110 [...] 0010	PC + 4 + Loop

Branch History Table (BHT)

2 state bits

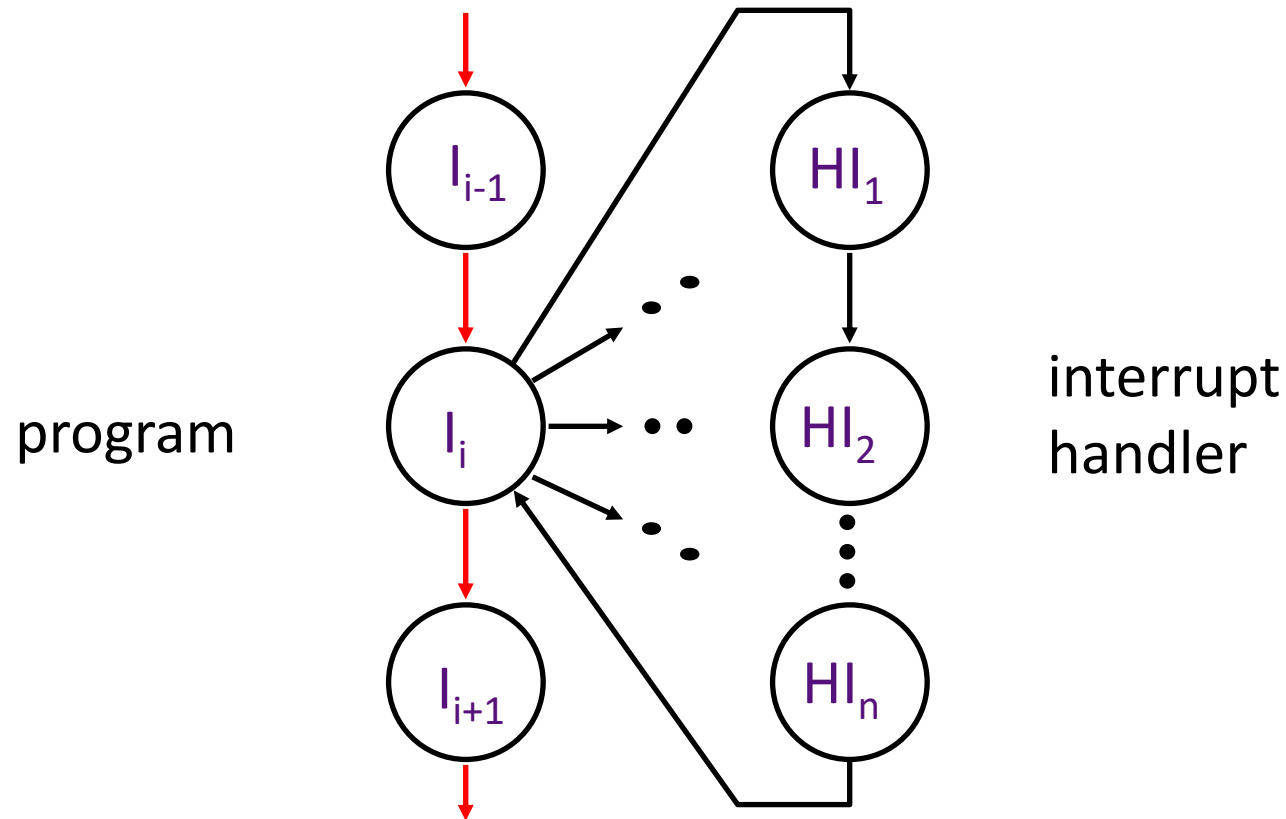
BTB is probed in the fetch stage along with the direction predictor.

A hit in the BTB means the PC is a branch PC.

Exception/Interrupt

An unscheduled event that disrupts program (instructions) in action.

Interrupt Handling

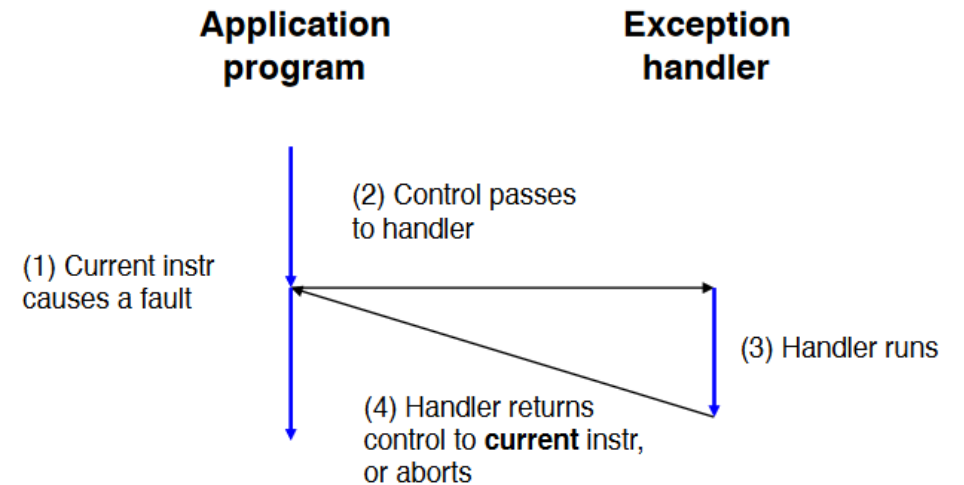
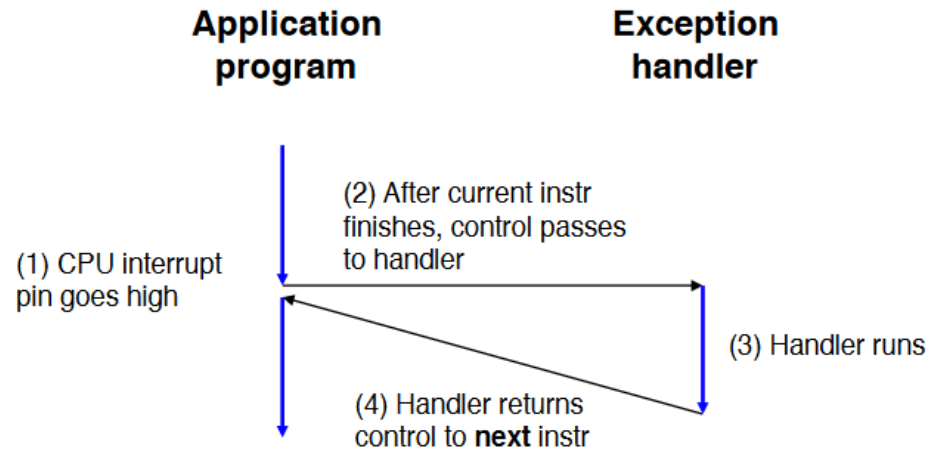


An *external or internal* event that needs to be processed. The event is usually unexpected or rare from program's point of view.

Causes

- Asynchronous: an *external event*
 - input/output device service-request
 - timer expiration
 - power disruptions, hardware failure
- Synchronous: an *internal event (a.k.a. traps or exceptions)*
 - undefined opcode, privileged instruction
 - arithmetic overflow, FPU exception, misaligned memory access
 - virtual memory exceptions*: page faults, TLB misses, protection violations
 - system calls, e.g., jumps into kernel

Interrupt and Exception



Interrupt Handler

Exception program counter (EPC):
address of the offending instruction,

Saves EPC before enabling interrupts
to allow nested interrupts

Need to mask further interrupts at
least until EPC can be saved

Need to read a *status register* that
indicates the cause of the interrupt

Handshake between processor and the OS

Processor:

stops the offending instruction,

makes sure all prior instructions complete,

flushes all the future instructions (in the pipeline)

Sets a register to show the cause

Saves EPC

Disables further interrupts

Jumps to pre-decided address (cause register or vectored)

Handshake between processor and the OS

OS:

Looks at the cause of the exception

Interrupt handler saves the GPRs

Handles the interrupt/exception

Calls RFE

Contd.

Uses a special indirect jump instruction RFE (*return-from-exception*) which

- enables interrupts
- restores the processor to the user mode



Coffee Credits

Lisan: +5

Tanay: +1



haben Sie einen
guten Tag