Oct 19th.

Note Title 19-10-2011

Recap.

Pipelines → data hazards
- → forwarding
- → stalling (LOAD-USE DEPENDENCY) LD → ST is a special case

control hazards
- → stall 2 cycles

→ branch prediction.

Branch Predictors.

Saturating Counter:

2-bit Sat Cntr.

v ← value of counter

inc: if (v < 3)
v++;

dec; if (v > 0)
v--;

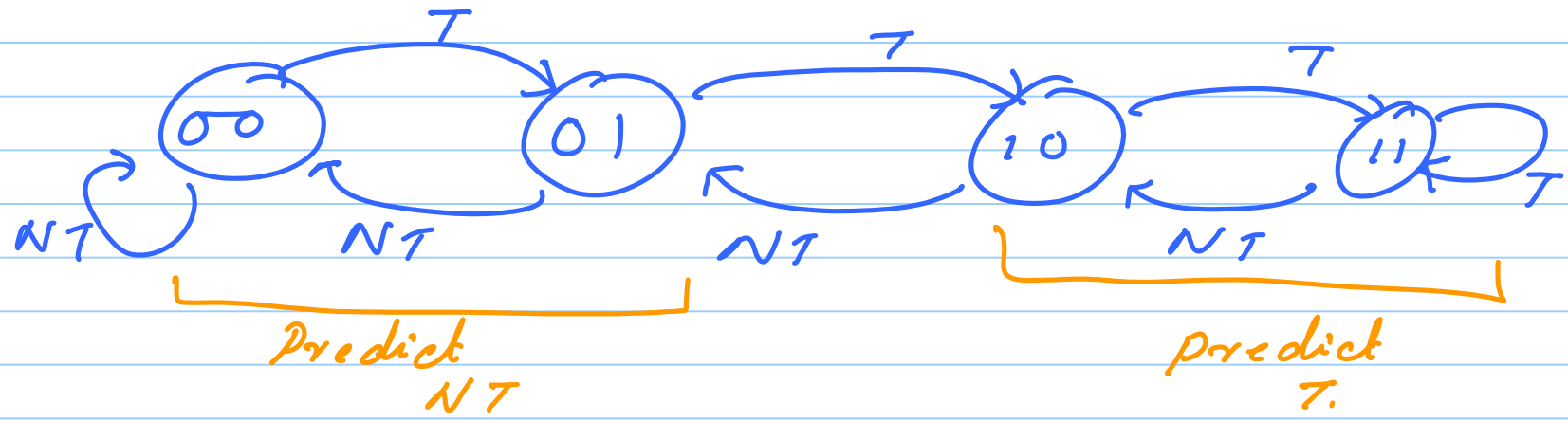
0 → NOT TAKEN
1 → TAKEN

Simplest Predictors

(1) Always Taken
(2) Always NOT TAKEN

Saturating Counter Predictor

One sat. cntr.

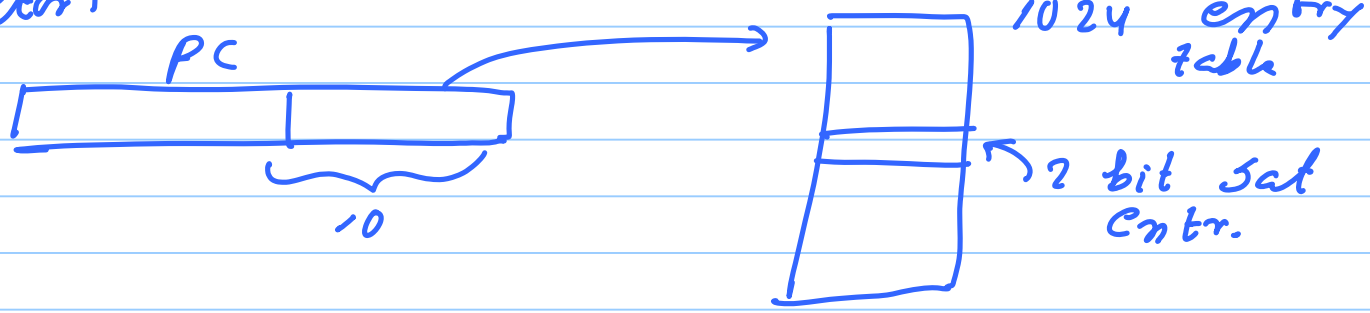


```
while (1) {
    if (world.is_falling_apart()) {
    }
}
```

T NT

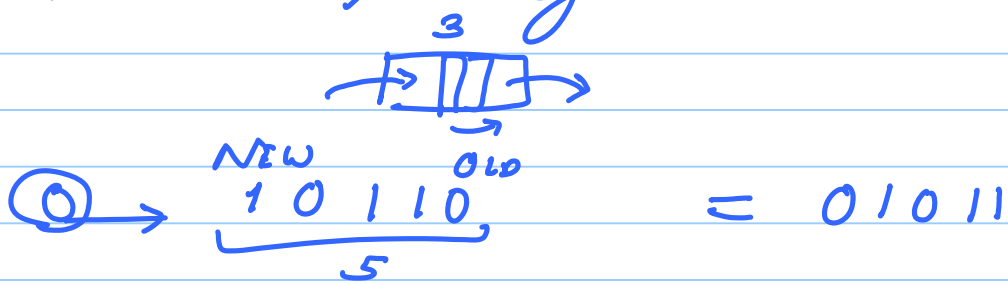
Separate Predictor for each PC.

Bimodal Predictor:



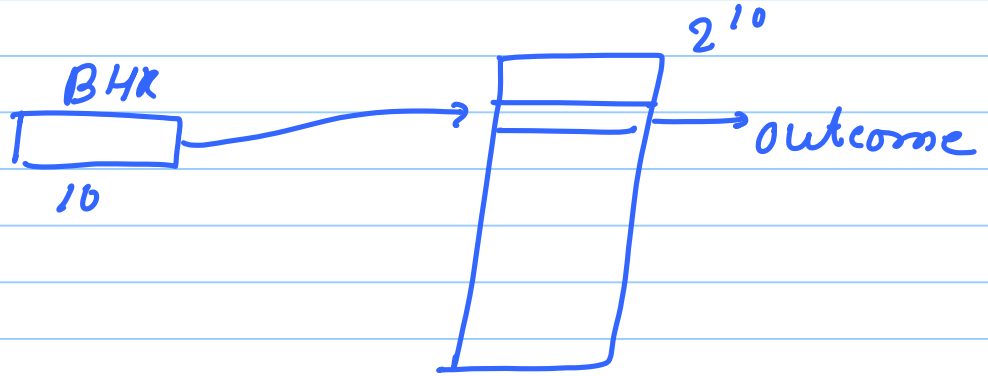
```
y = 0 ;
if (x == 1)
        y = 1;
if (y == 1)
        printf ("great ");
```

Branch History Register (BHR)

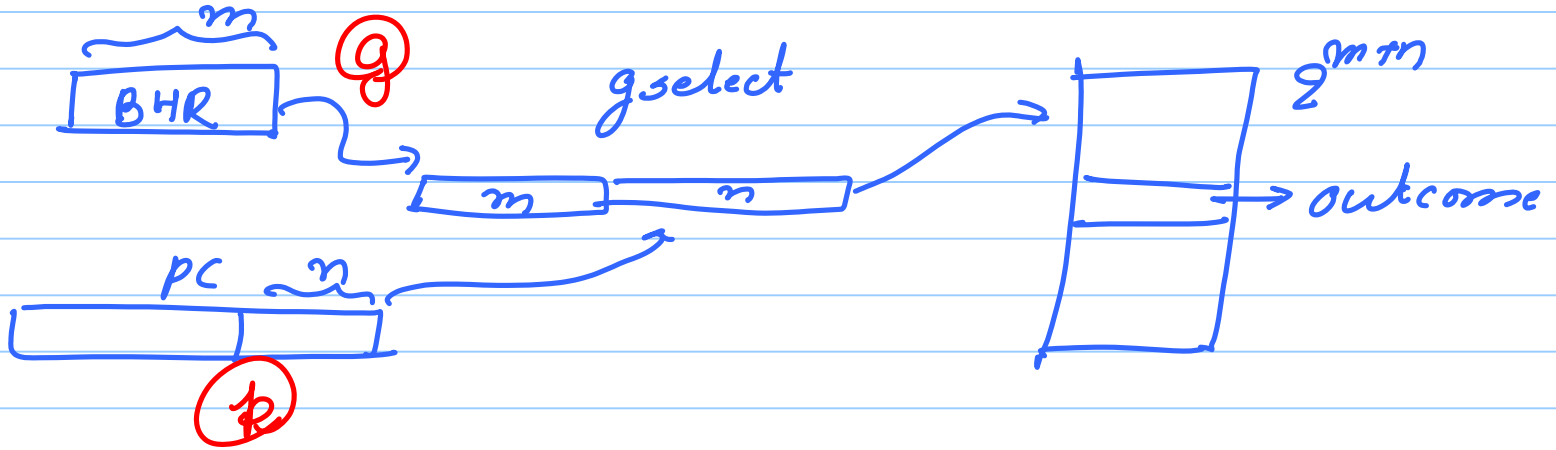


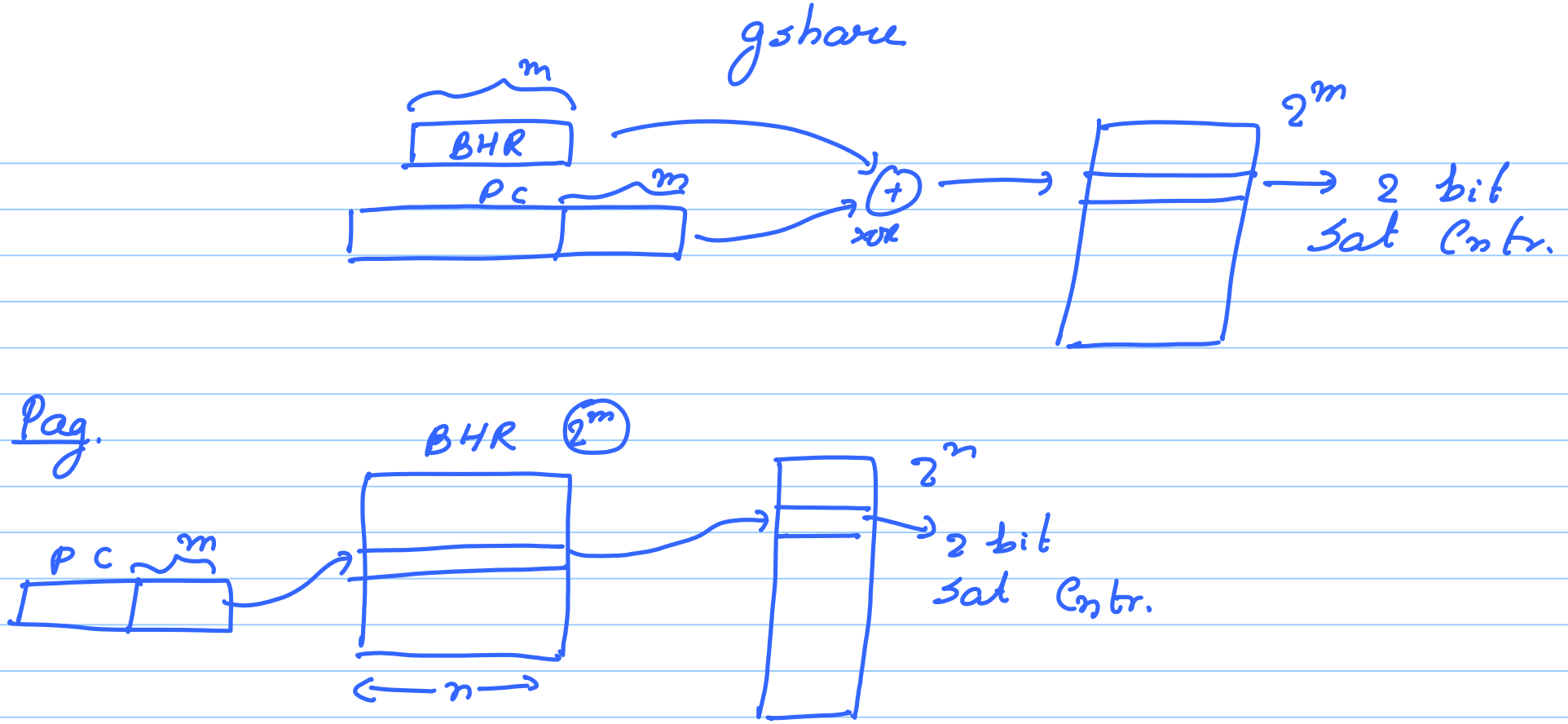
Family of Predictors

Gag.

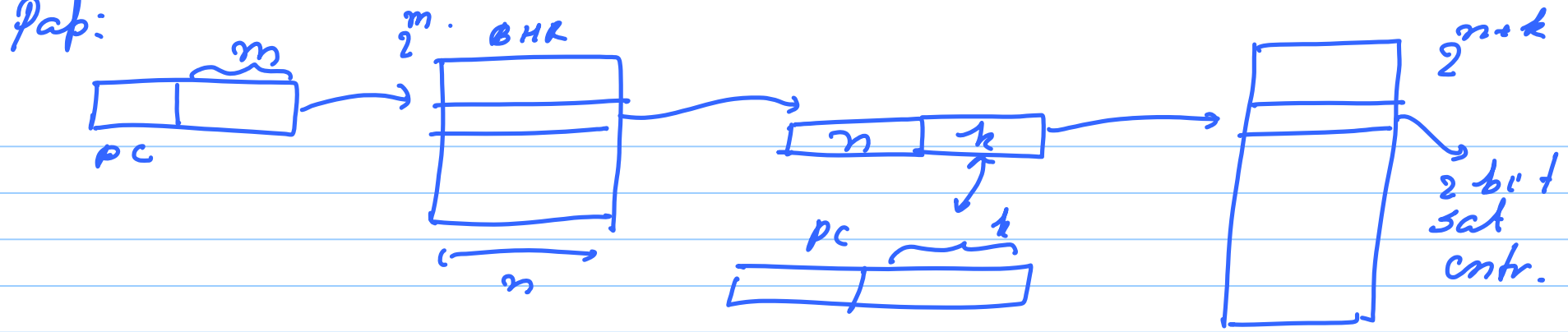


Gap:

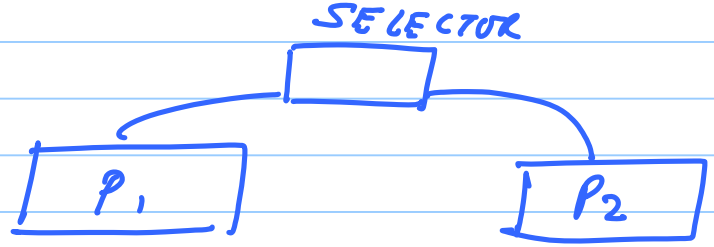


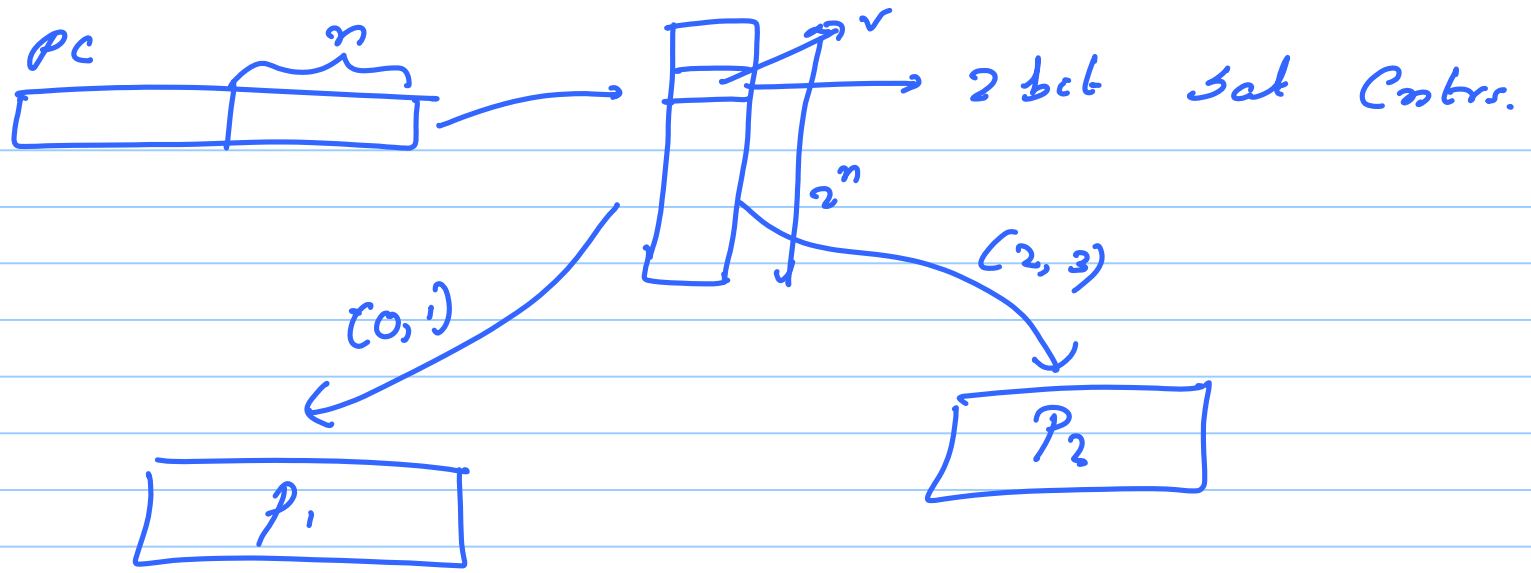

gshare
m
BHR
PC
m
XOR
2^m
2 bit
Sat Cntr.
Pag.
BHR
2^m
PC
m
n
2^n
2 bit
Sat Cntr.

Pap:



Tournament:





Both predictors are correct : DO NOTHING

" " " wrong : DO NOTHING

P_1 ✓ P_2 ✗ : dec (v)

P_2 ✓ P_1 ✗ : inc (v)

Prediction: 1) Check the selector
2) Go to appropriate Predictor.
3) Predict

Training:
1) Train both predictors
2) Train the selector.

Advanced Pipelining

1) Exceptions
2) Multiple Issue

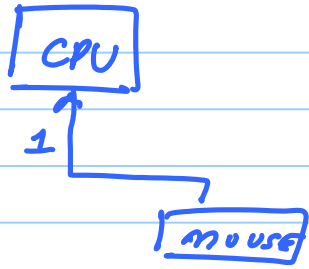
Exceptions

Internal

Erroneous program execution

1) (÷ by 0)
   (illegal mem. access)

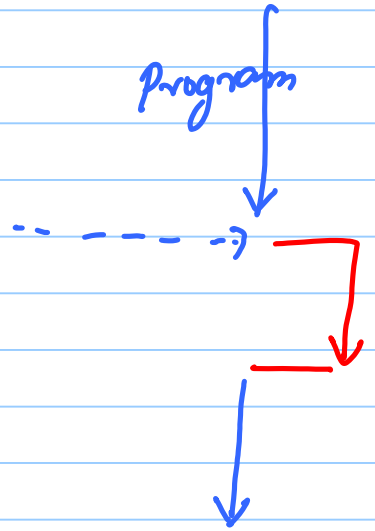
External



1) Device raises a request.
2) CPU would finish immediate work.
3) Consider the request.

2) Process this information.

4) Process the interrupt

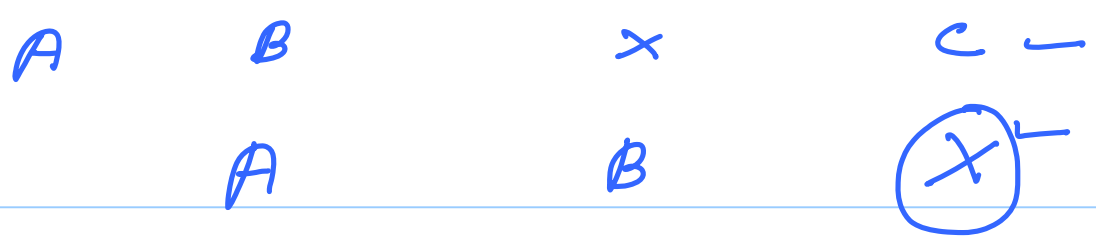


Precise Exception:

a) All instructions after the interrupt don't complete

b) Before the interrupt all instructions complete

IF	ID	EX	MEM	WB
A	B	(÷0) (X)	C	D



1) When X enters the WB stage
2) Cleanup the pipeline
3) HW Saves the registers and next pc
4) Load the interrupt / exception handler
5) Restore the pc and registers

Special case:
B is a store

Soln-1 Before sending a store — check.

Soln.2 When x passes through the MEM stage — it disables it.

```
try {
        A[0] = 0;
        x = 0/0;
        A[0] = 1;
catch (Exception EX) {
        print (A[0]);
}
```

Is 1 a valid outcome! NO

Next Class:

1) Multiple Issue.

2) Caches.