Computer Architecture: Fundamentals, Tradeoffs, Challenges

Chapter 4: Because of Pipelines (out-of-order execution, branch prediction)

Yale Patt

The University of Texas at Austin

Austin, Texas

Fall, 2022

Outline

- Out-of-order Execution
  - The Tomasulo Algorithm (the IBM 360/91)
    - Pipelining with a hiccup
    - An example
  - What was wrong with it (did not have precise exceptions)
  - How do we fix it (the Reorder Buffer)
    - Pipelining with a second hiccup
- Branch Prediction
  - Other ways to deal with conditional branches
    - Predication (the THUMB IT instruction)
    - Multiple pipelines
    - Delayed branch
    - Combine branches ← RS 6000
  - Compile time branch prediction (BTFN, hint bits)
  - Runtime (LT, 2bitCtr, Two-level, Hybrid, Perceptron, TAGE)
  - The HEP (Burton Smith, Donelcor (~1978)

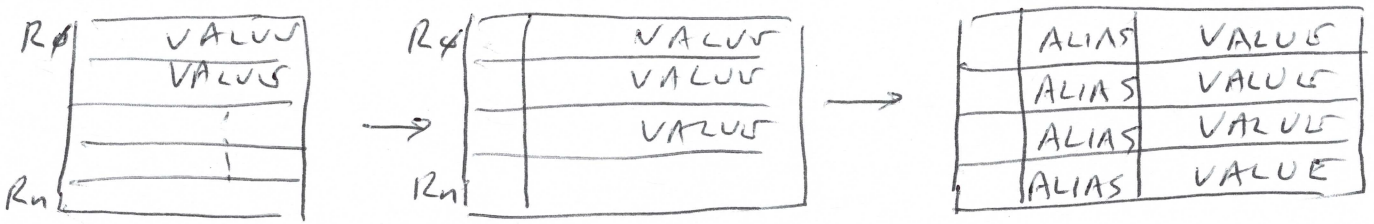
THE SCOREBOARD HAS A PROBLEM

MANY INSTRUCTIONS ARE UNNECESSARILY HELD UP.

TRAFFIC ANALOGY: CARS THAT ARE BLOCKED

HOW ~~DO~~ TO DEAL WITH THIS BLOCKING?

ANSWER: GET THEM OUT OF THE WAY



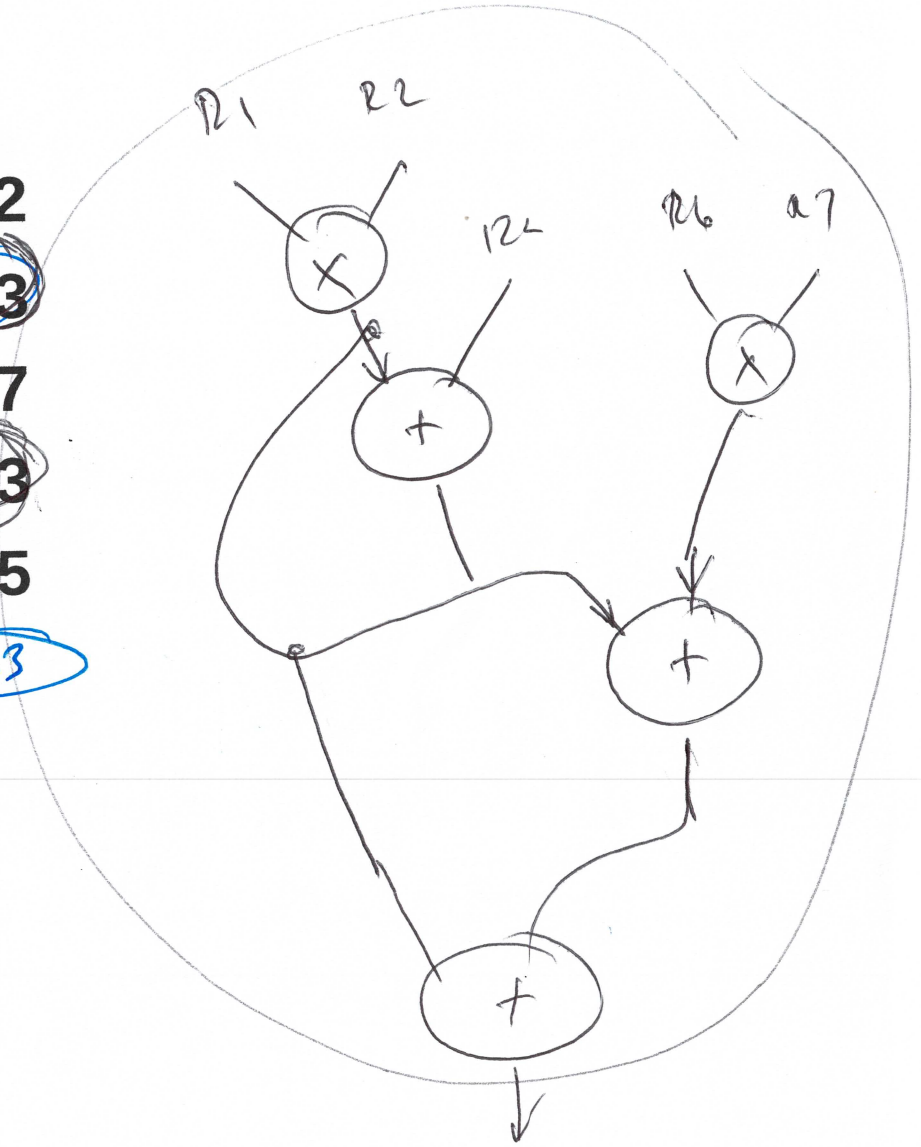
SCOREBOARD

~~AND~~ A ~~RE~~-ORDER BUFFER ~~ROB~~

An example

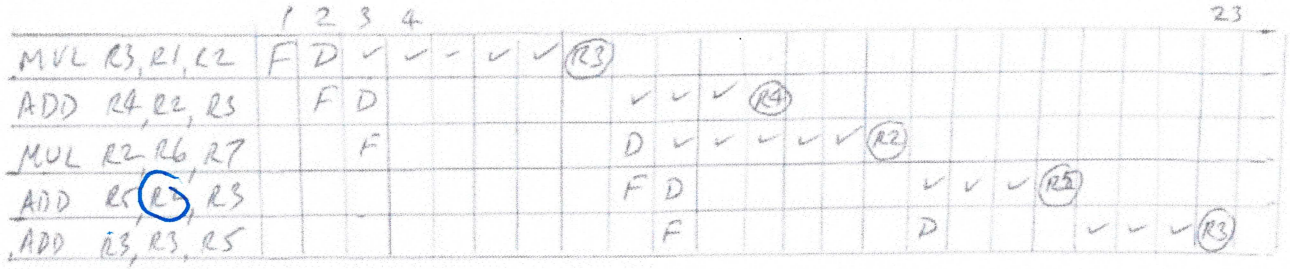
```
MUL R3,R1,R2
ADD R4,R2,R3
MUL R2,R6,R7
ADD R5,R2,R3
ADD R3,R3,R5
```

MUL X, Y R3

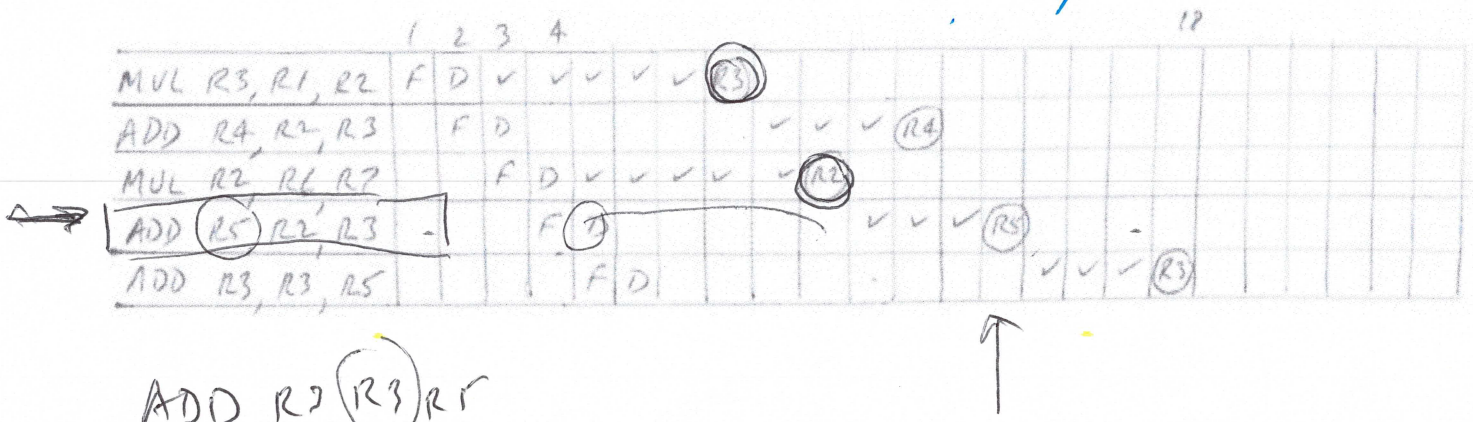


Out-of-Order execution (example)

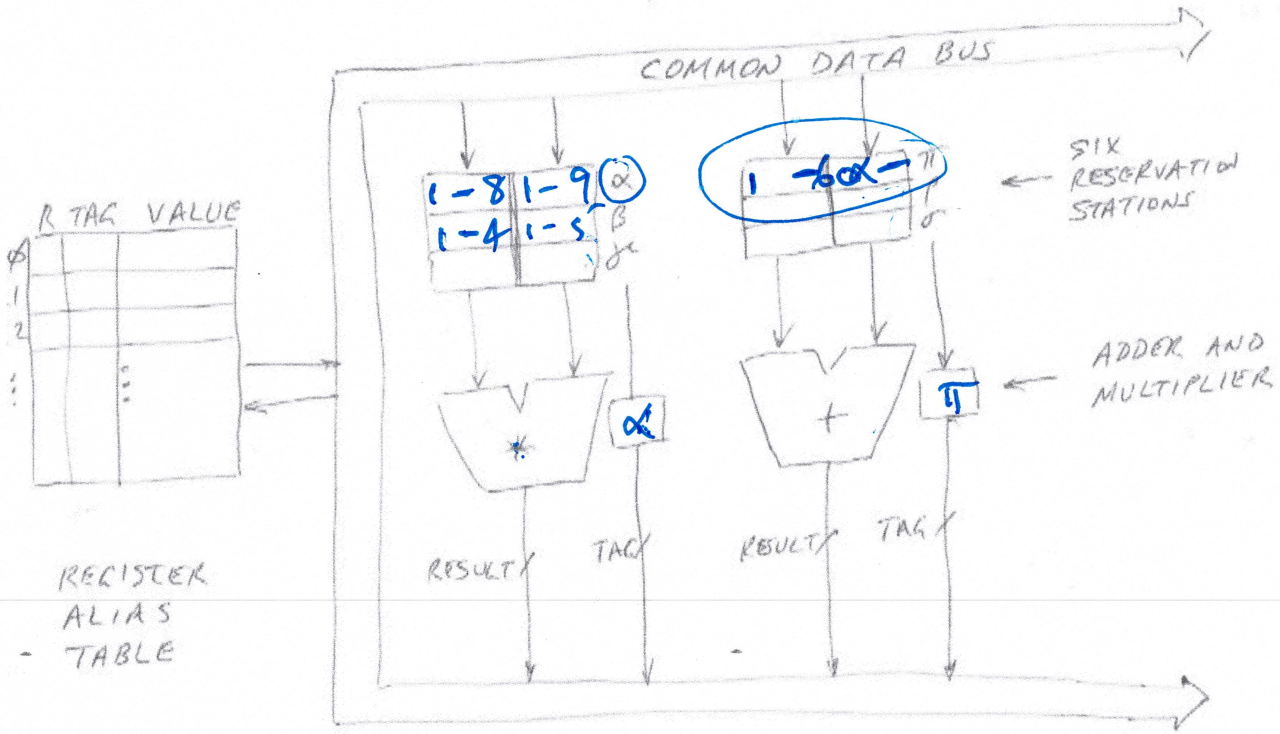
- **In-order**

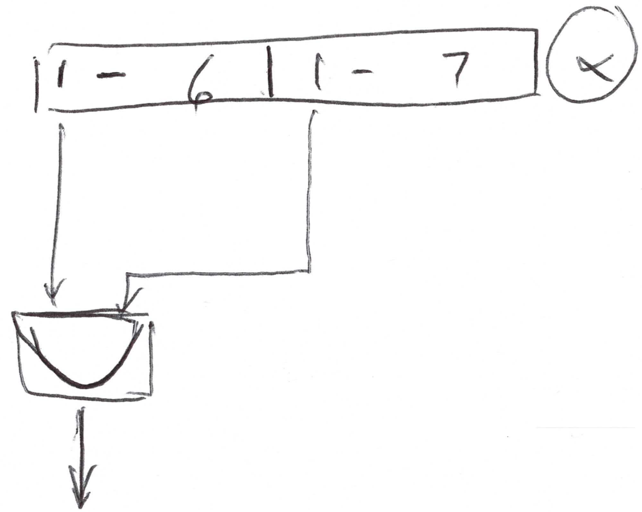


- **Out-of-Order**

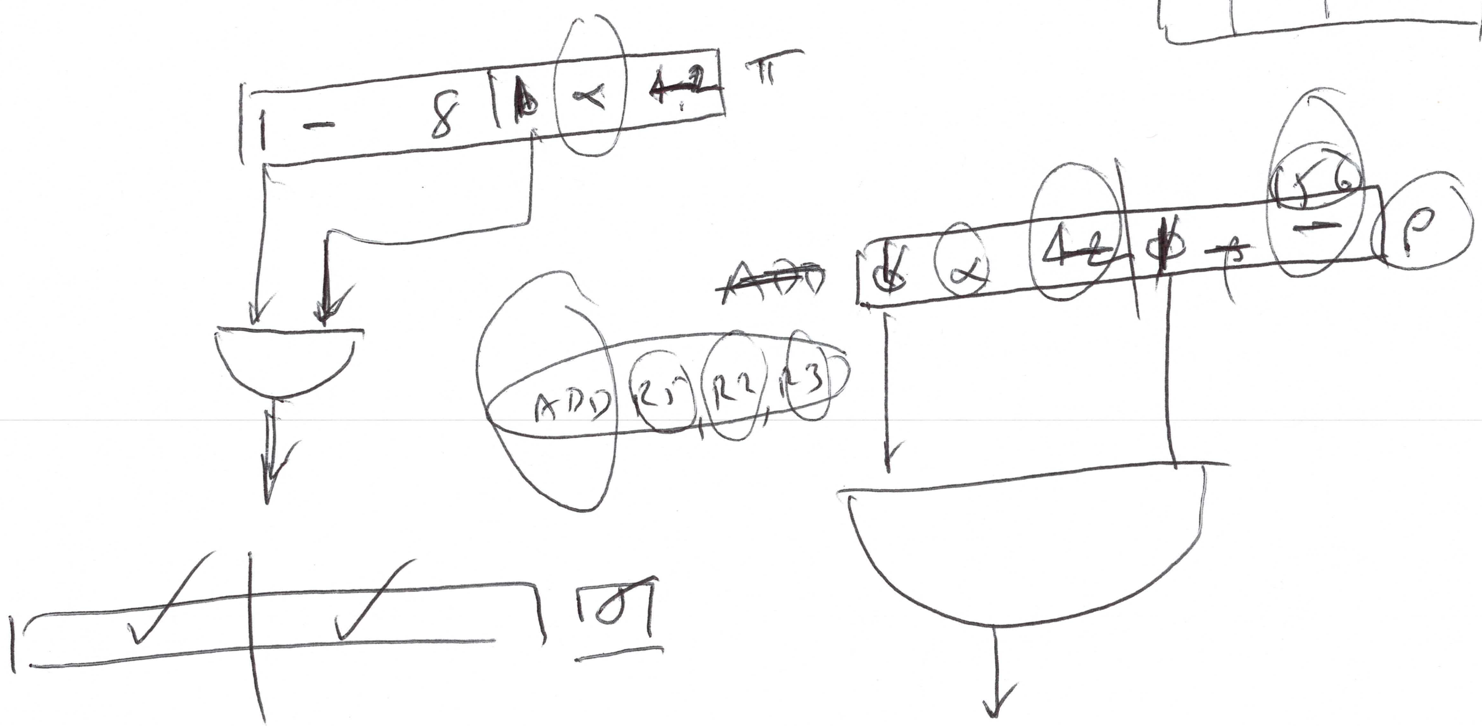


Tomasulo Algorithm Microarchitecture





R0	1	–	6
R1	1	–	7
R2	1	–	8
R3	0	α	9
R4	1		10
R5	0		198
R6	1		12
R7	1		13



Out-of-order commit violates precise exceptions

- **Precise exceptions**
  - Identity of the faulting instruction is known
  - State recovered to just before faulting instruction
- **A simple example:**

DIV R1,R2,R3

ADD R2,R3,R4

ADD takes one cycle and stores result in R2

DIV eventually takes an exception.

Not possible to recover R2 prior to DIV execution

How To Fix The Problem

- Add a structure (Reorder Buffer)
- Fetch An Instruction
  Allocate An Entry

Code

I_1: DIV R1, R2, R3
I_2: ADD R2, R3, R4
:

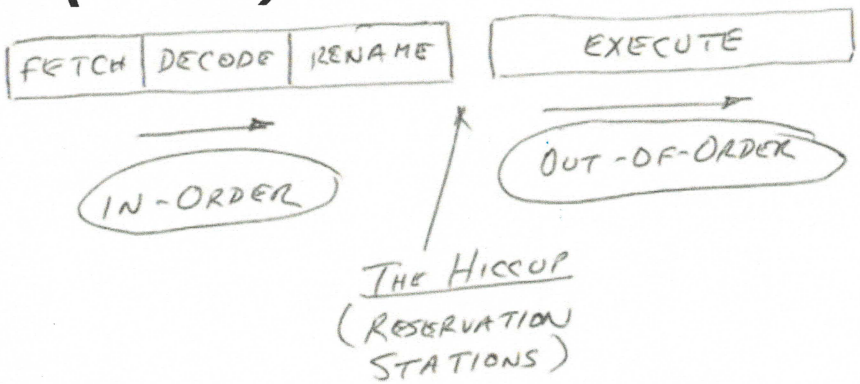
For each instruction,

(1) Result of Execution Is Temporarily Stored In The Reorder Buffer (NOT In The Register)

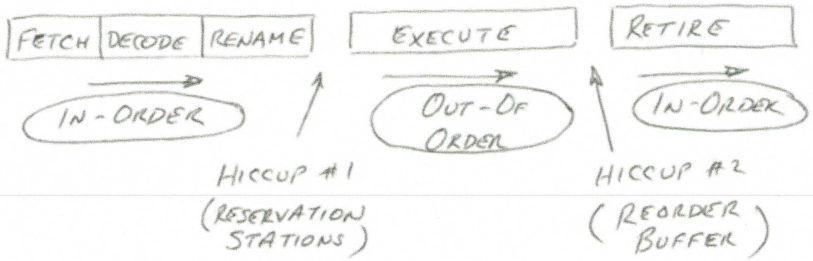
(2) The Result Is Stored In The Register After All Older ~~[illegible]~~ Results Are Stored In The Registers.

Pipelining with “hiccups”

- **Tomasulo (~1965)**



- **HPS (~1985)**

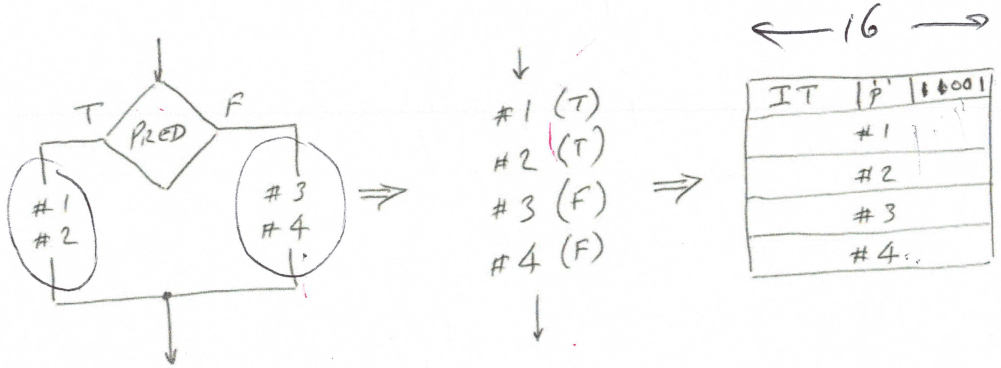


Branch Prediction

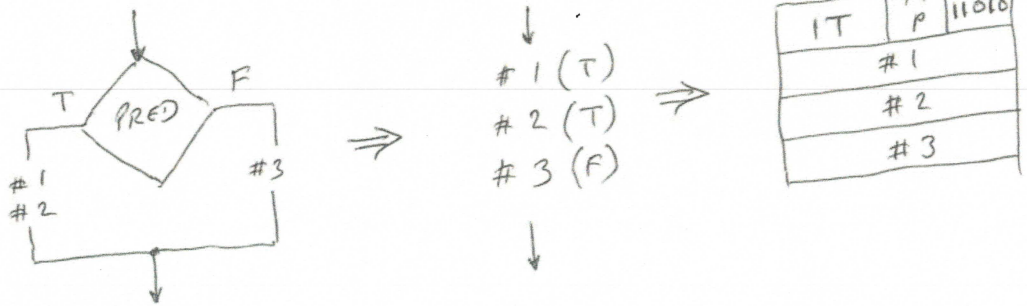
- Ways to deal with Conditional Branches
  - Compile time
    - BTFN
    - Hint bits
    - Predication (the THUMB IT instruction)
    - Delayed branch
    - Combine branches
  - Run time Branch Prediction
    - LT
    - 2bit Counter
    - Two-level
    - Hybrid
    - Perceptron
    - TAGE
- The HEP (Burton Smith, Donelcor (~1978)

Predicated Execution (THUMB ISA)

- **Example 1**
  (4 instructions)



- **Example 2**
  (3 instructions)

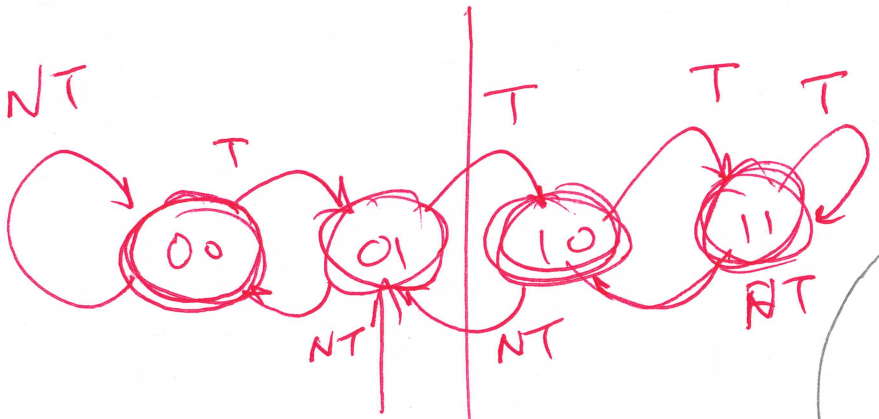


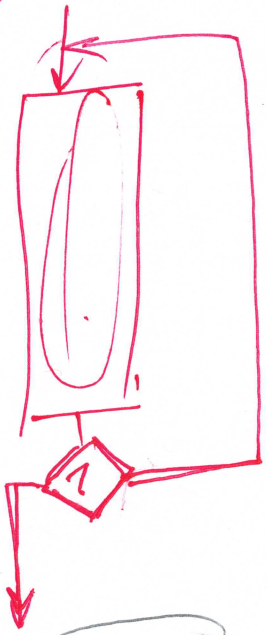
COMPILE TIME → BTFN

RUN-TIME:

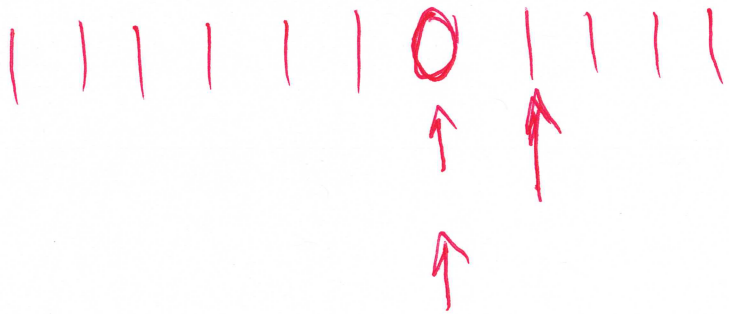
- LT
- 2-bit Counter

BR A





ISCA
1981
JIM SMITH

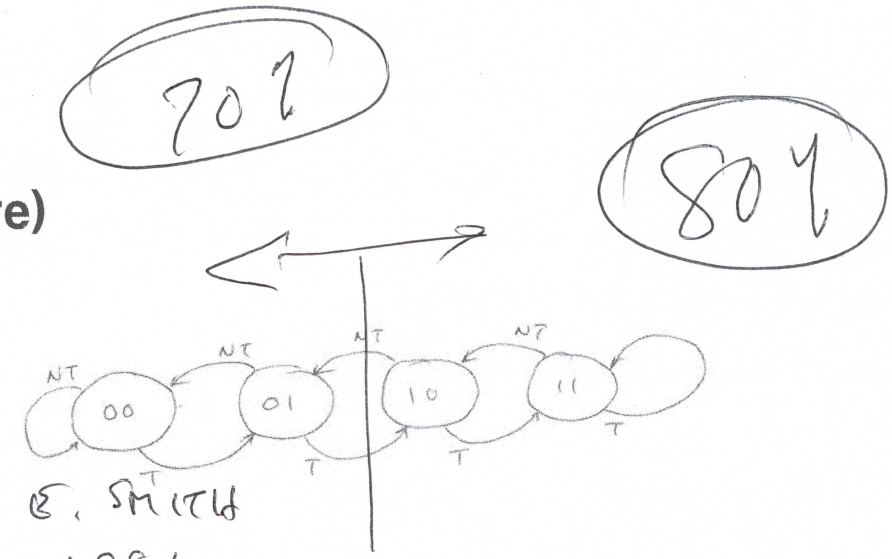


n-1
n-2

3
2

Dynamic branch prediction

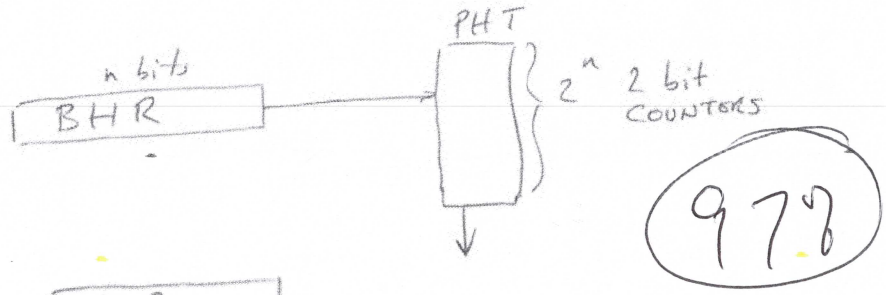
- **Last time taken**
  (Store a bit in the I Cache Tag Store)



- **Saturating 2 bit counter**
  (Use high bit to predict)

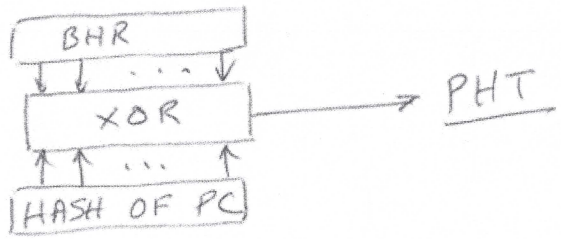
JAMES E. SMITH
ISCA 1981

- **Two-level Predictor**



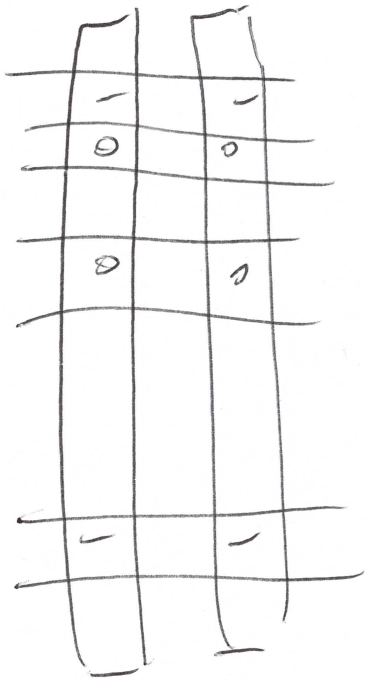
- **Gshare Modification**

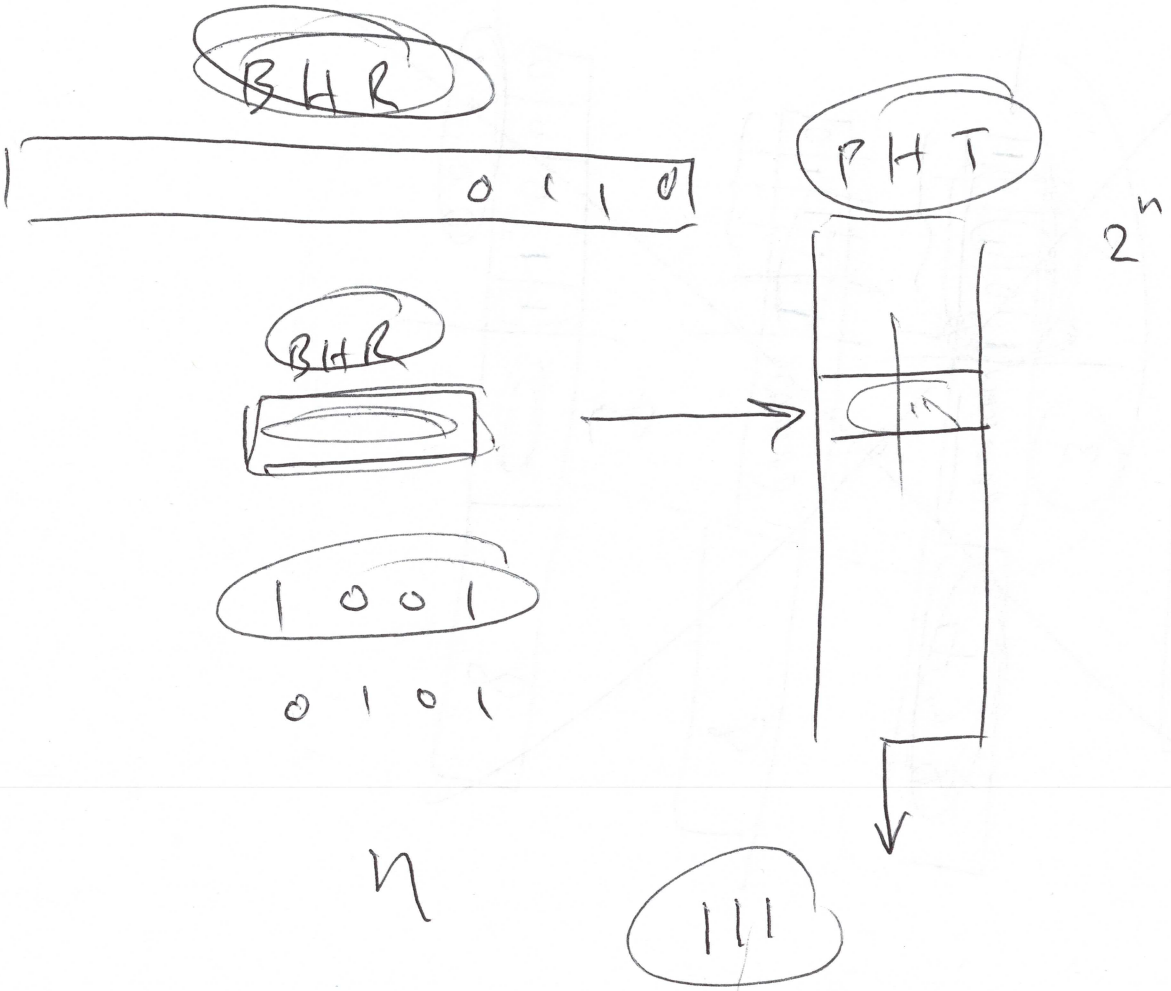
SCOTT McFARLING



Two-Level Adaptive Branch Prediction

The Basic Idea





TSE-YU YEH

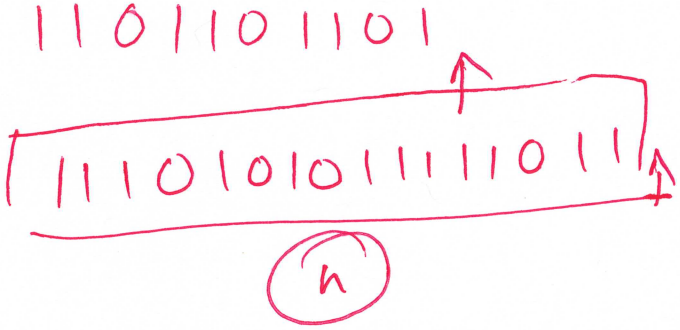
MICRO

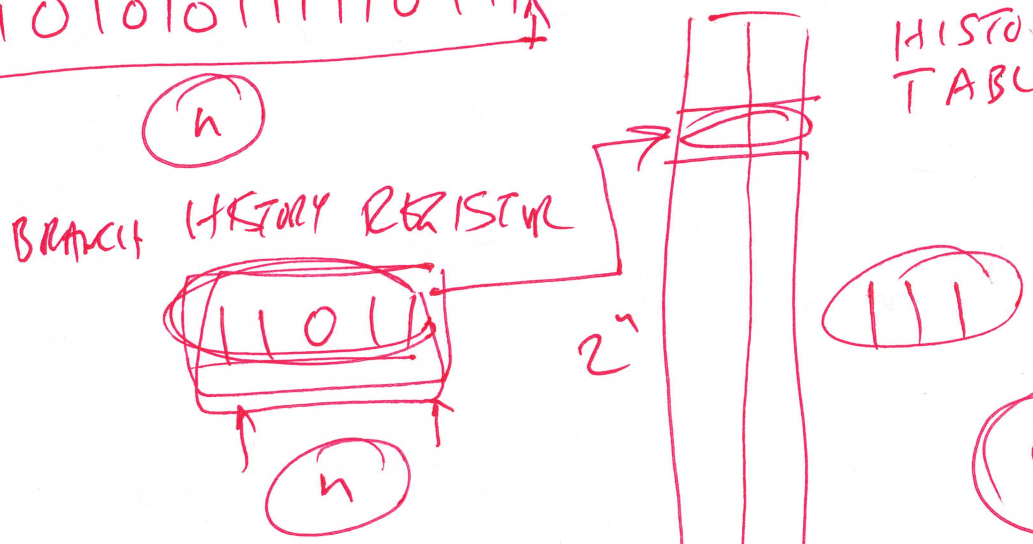
1991

INTEL PENTIUM PRO

TAGE + BRANCHNET - ANDRE SEZNEC

SIAVASH ZANGENEH

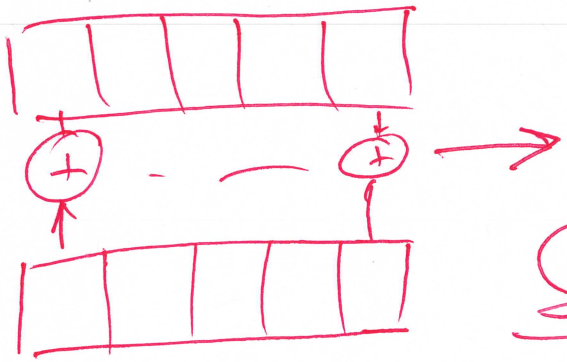




SCOTT McFARLING

1993

97%



GSHARE

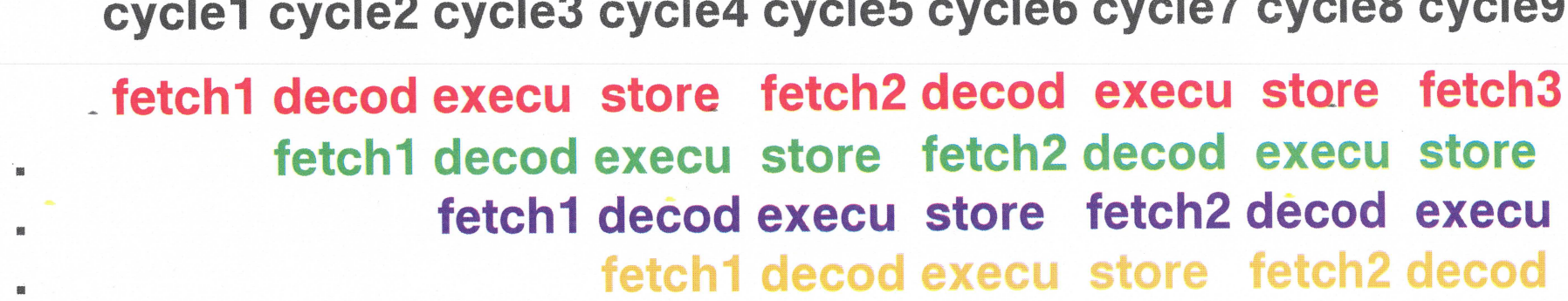
KIMMING SO

IBM

ASPLOS 1992

The Heterogeneous Element Processor (HEP)

- Burton Smith (at Donelcor ~1975)
- Introduction of Multithreading
  - Burton always credited CDC6600; I credit Burton!
- The mechanism (modified for explanation)
  - Four stages, HEP had 8
  - Four threads in progress, HEP had 8
  - Not shown: the hopper (20 threads waiting to participate)
    - *IF a thread's instruction could not finish in time for next instruction*



THE HEP CONCEPT (THE BIRTH OF MULTITHREADING)

THIS EXAMPLES: 4 STAGES, NOT 8
4 THREADS ACTIVE, NOT 8

CYCLE	1	2	3	4	5	6	7	8	9	10	11	...
	F_1	D_1	EX_1	ST_1	F_2	D_2	EX_2	ST_2	F_3	D_3	EX_3	...
		$\mathcal{F}_1$	$\mathcal{D}_1$	$\mathcal{E}_1$	$\mathcal{S}_1$	$\mathcal{F}_2$	$\mathcal{D}_2$	$\mathcal{E}_2$	$\mathcal{S}_2$	$\mathcal{F}_3$	$\mathcal{D}_3$	...
			f_1	d_1	e_1	s_1	f_2	d_2	e_2	s_2	f_3	...
				ϕ_1	ϑ_1	ϵ_1	σ_1	ϕ_2	ϑ_2	ϵ_2	σ_2	...

Xie xie!