Pipeline Control Hazards and Instruction Variations

Hakim Weatherspoon
CS 3410, Spring 2011
Computer Science
Cornell University

See P&H Appendix 4.8 & 2.16 and 2.17

Announcements

PA1 available: mini-MIPS processor

PA1 due next Friday

Work in pairs

Use your resources

- FAQ, class notes, book, Sections, office hours, newsgroup, CSUGLab

Prelims:

- Thursday, March 10th in class
- Thursday, April 28th Evening

Goals for Today

Recap: Data Hazards

Control Hazards

- What is the next instruction to execute if a branch is taken? Not taken?
- How to resolve control hazards
- Optimizations

Instruction Variations

- ARM
- x86

Data Hazard Recap

Delay Slot(s)

- Modify ISA to match implementation

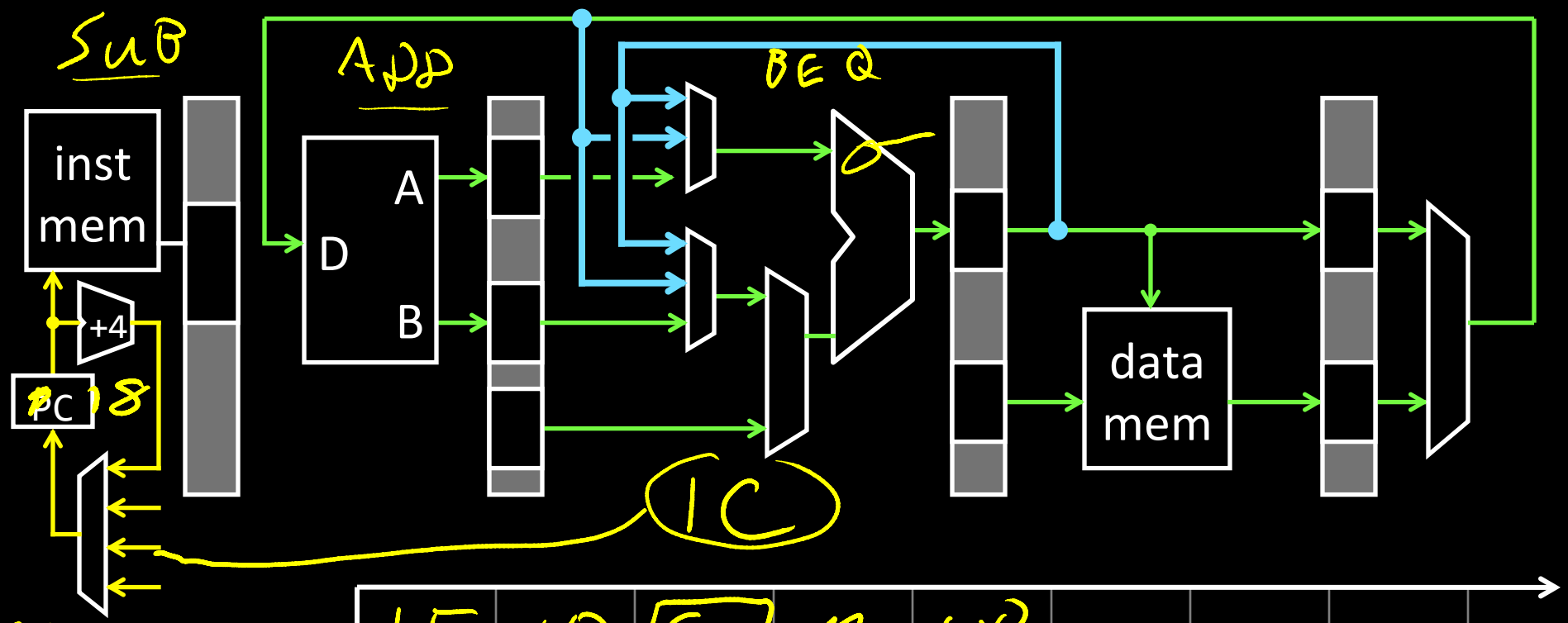
Stall

- Pause current and all subsequent instructions

Forward/Bypass

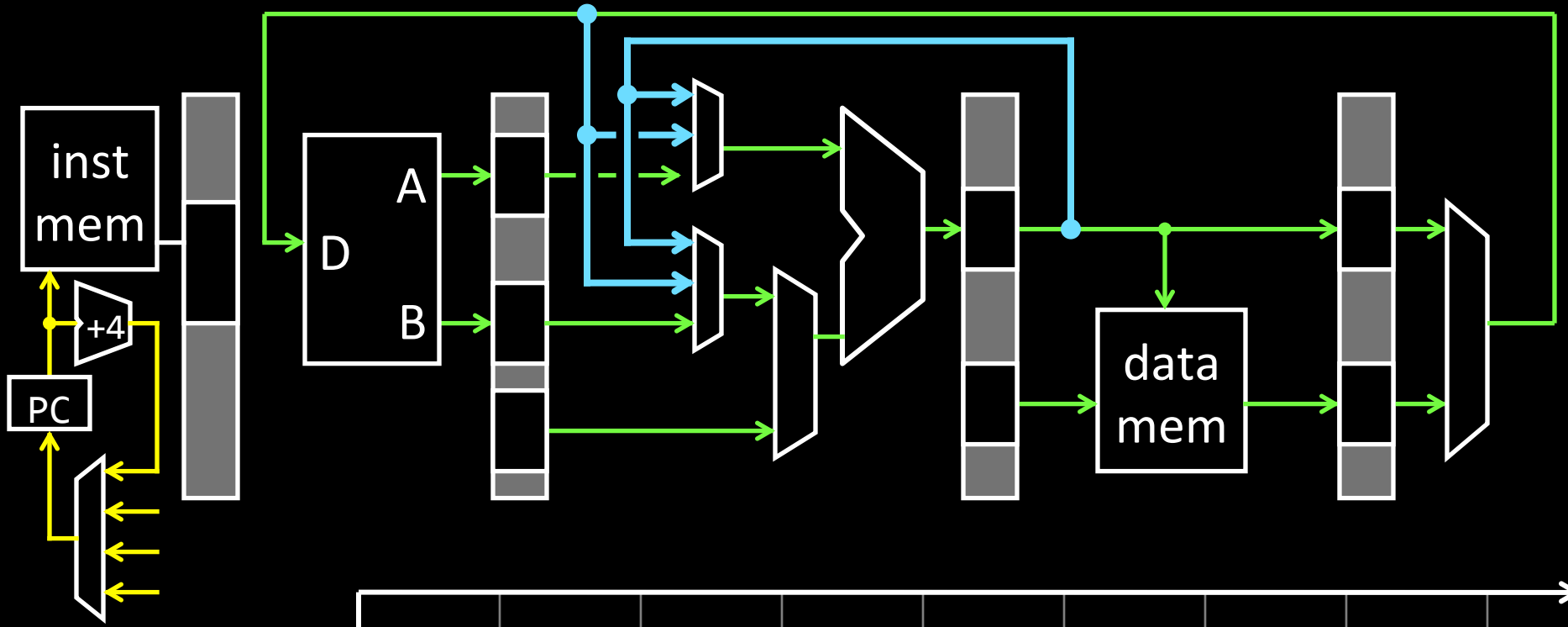
- Try to steal correct value from elsewhere in pipeline
- Otherwise, fall back to stalling or require a delay slot

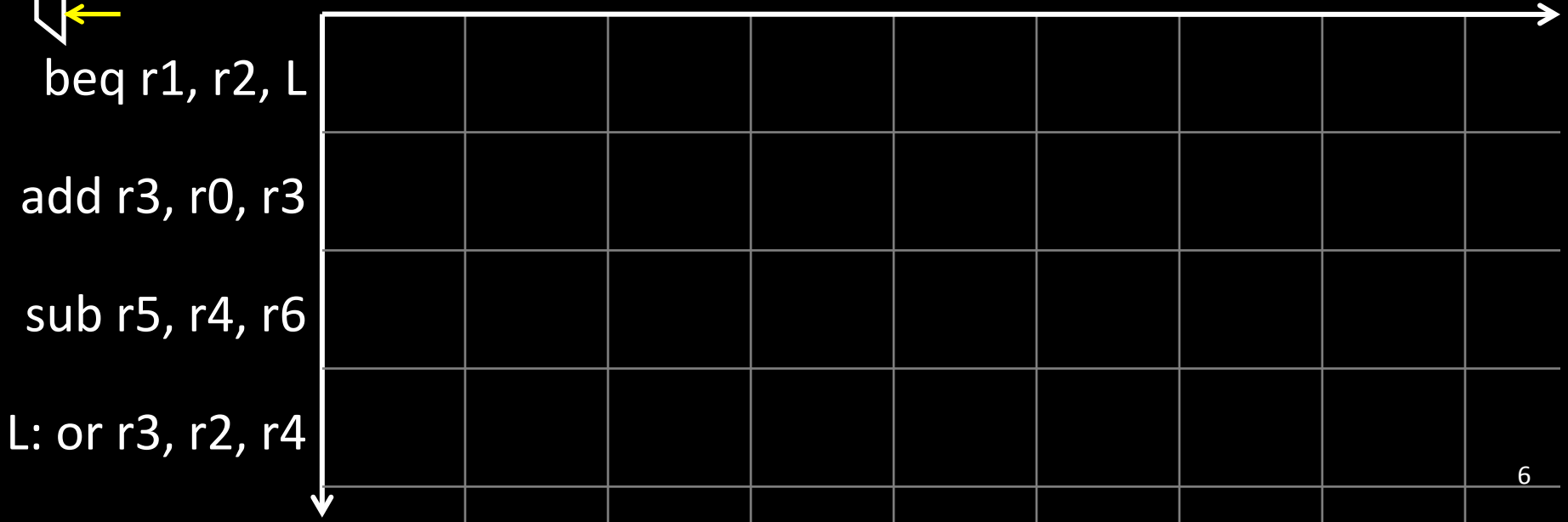
More Hazards



10 beq r1, r2, L	IF	ID	EX	M	WB				
14 add r3, r0, r3		IF							
18 sub r5, r4, r6			IF	NOP					
1C L: or r3, r2, r4				IF					

More Hazards





Control Hazards

Control Hazards

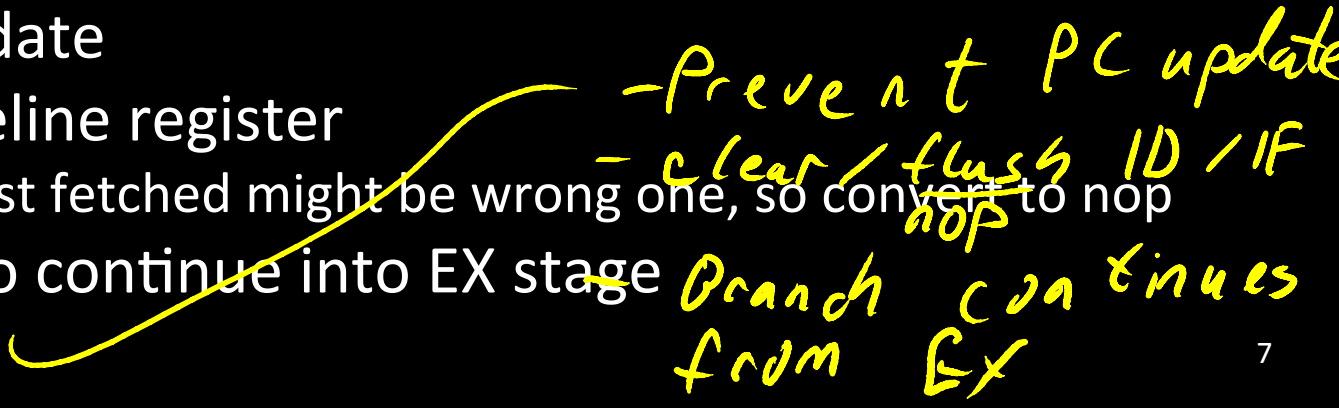
- instructions are fetched in stage 1 (IF)
- branch and jump decisions occur in stage 3 (EX)
- i.e. next PC is not known until 2 cycles after branch/jump

Delay Slot

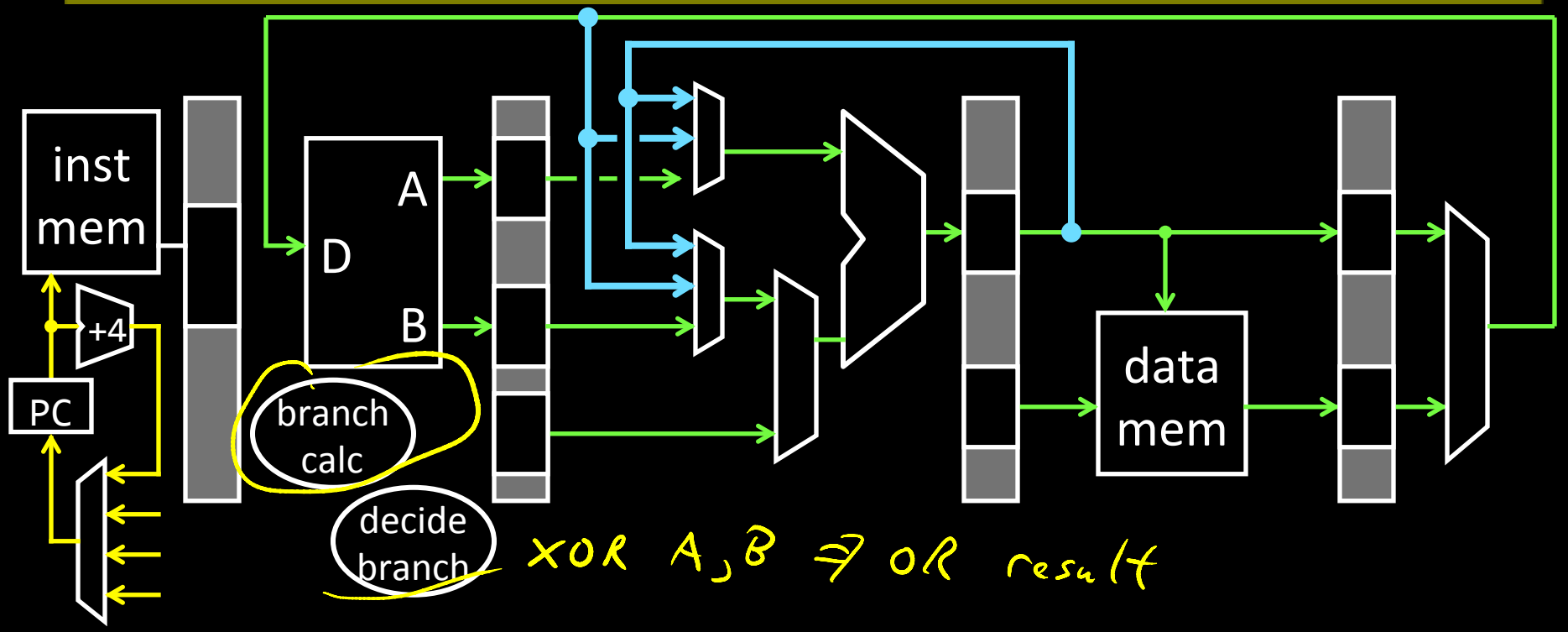
- ISA says N instructions after branch/jump always executed
  - MIPS has 1 branch delay slot

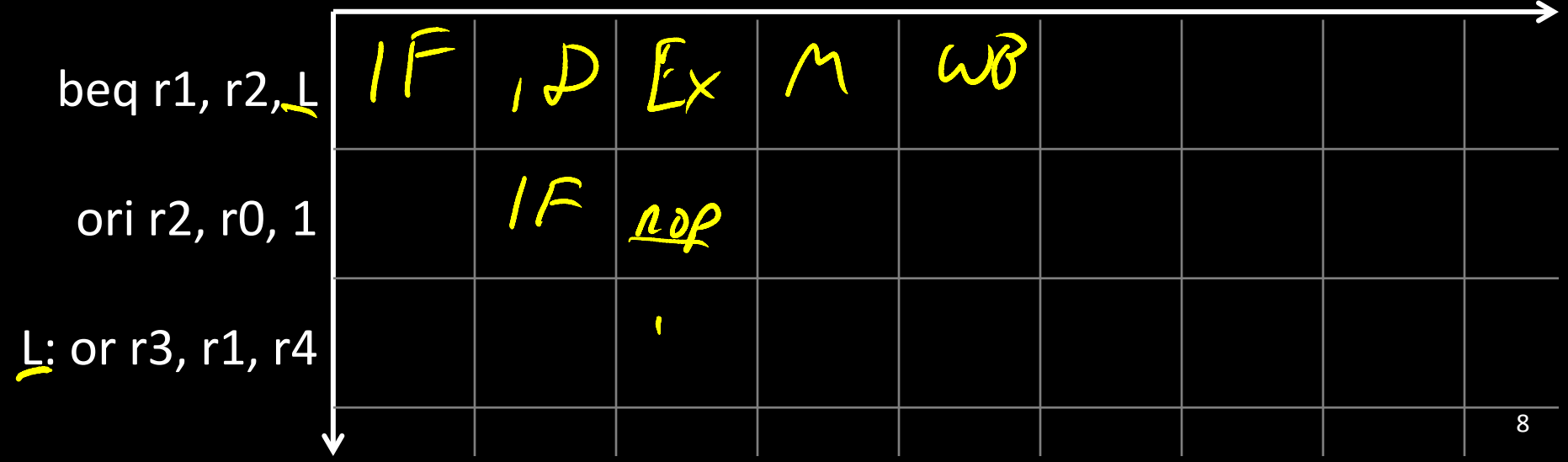
Stall (+ Zap)

- prevent PC update
- clear IF/ID pipeline register
  - instruction just fetched might be wrong one, so convert to nop
- allow branch to continue into EX stage



Delay Slot


inst mem
+4
PC
A
D
B
branch calc
decide branch
data mem
XOR A, B ⇒ OR result


beq r1, r2, L
ori r2, r0, 1
L: or r3, r1, r4
IF
ID
EX
M
WB
IF
nop

Control Hazards: Speculative Execution

Control Hazards

- instructions are fetched in stage 1 (IF)
- branch and jump decisions occur in stage 3 (EX)
- i.e. next PC not known until 2 cycles after branch/jump

Stall

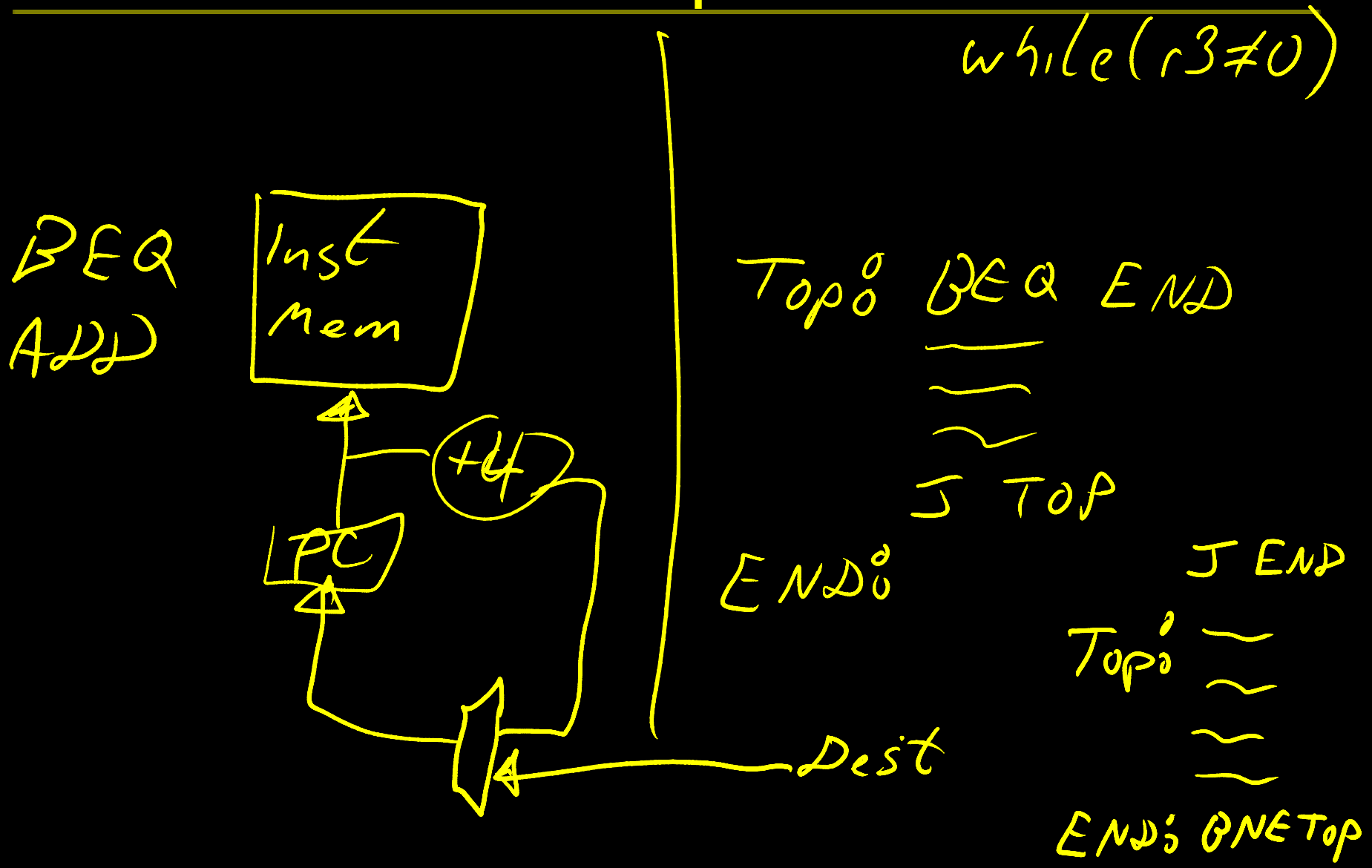
Delay Slot

Speculative Execution

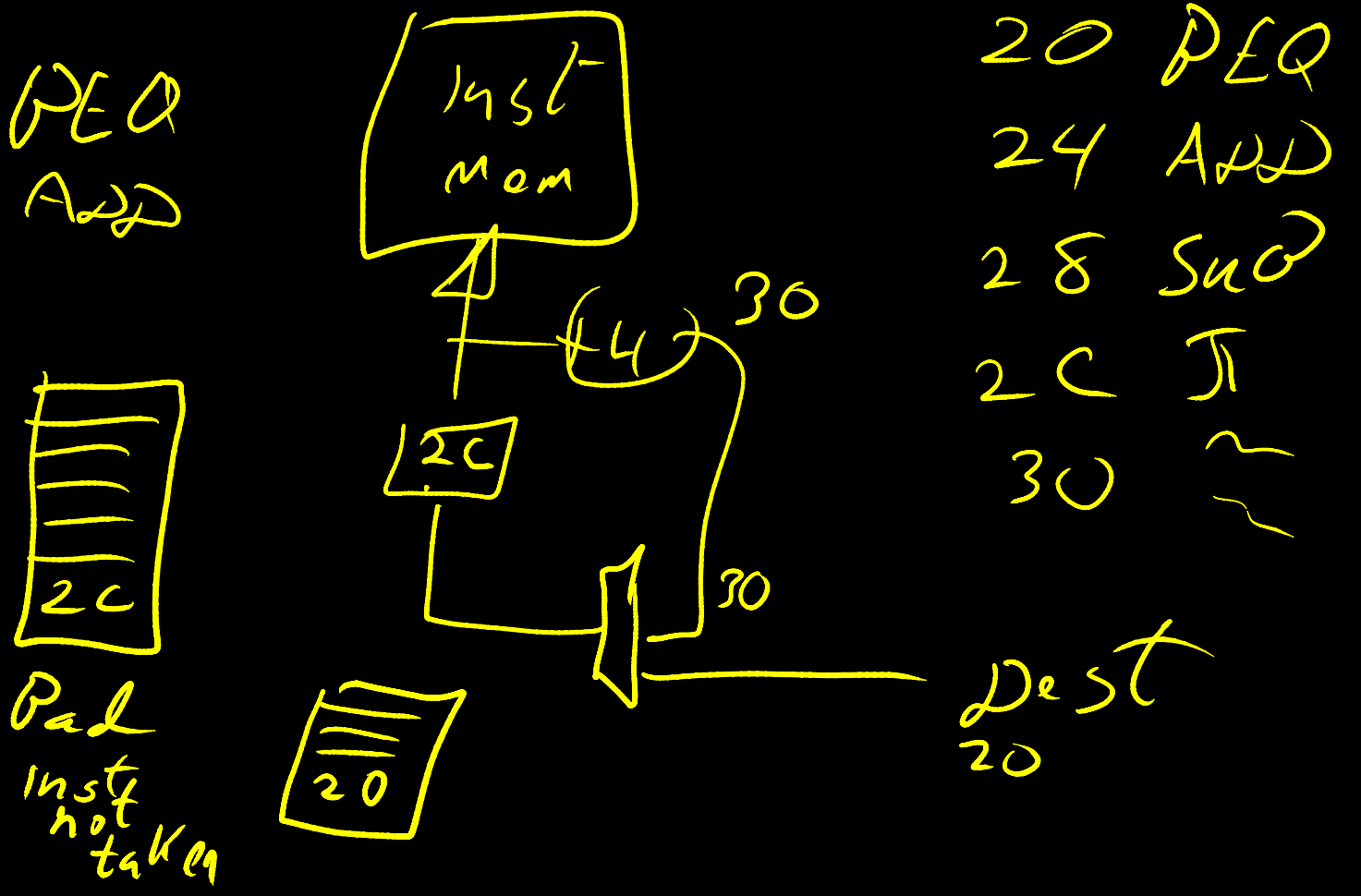
- *"Guess"* direction of the branch
  - Allow instructions to move through pipeline
  - Zap them later if wrong guess
- Useful for long pipelines

dynamic branch prediction

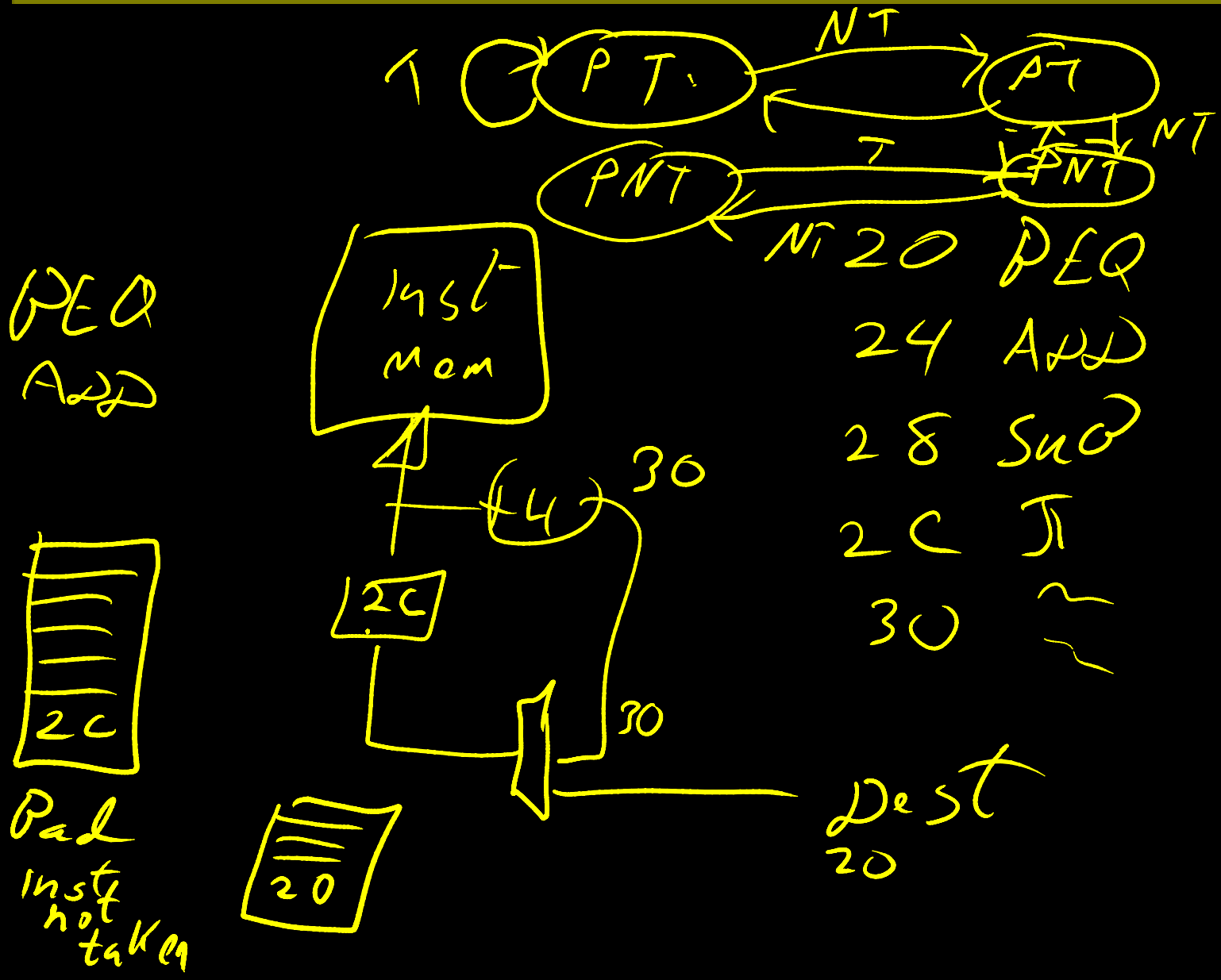
Loops


while(r3≠0)
BEQ
ADD
Inst
Mem
+4
PC
Top: BEQ END
J TOP
END:
J END
Top'
END: BNE Top
Dest

Branch Prediction



Branch Prediction



Pipelining: What Could Possibly Go Wrong?

Data hazards

- register file reads occur in stage 2 (IF)
- register file writes occur in stage 5 (WB)
- next instructions may read values soon to be written

Control hazards

- branch instruction may change the PC in stage 3 (EX)
- next instructions have already started executing

Structural hazards

- resource contention
- so far: impossible because of ISA and pipeline design

Goals for Today

Recap: Data Hazards

Pg 162

Control Hazards

- What is the next instruction to execute if a branch is taken? Not taken?
- How to resolve control hazards
- Optimizations

Advanced RISC Machine

Instruction Variations

- ARM
- x86

Armdroid = Android ARM

Winted = Windows Intel

① same yr RISC MIPS 1985

MIPS vs ARM
3 inst 9
32 reg 16
r0

Variations

More shifts

Shift one reg (C) any amt
add to another reg (B)
store result in another (A)

ARM

$A = B + x * C$

$A = B + LI * C$

Variations

Condition flags

ARM

if (A < B) goto L

MIPS

Sub A - B

Branch (B - A) less than 0

Sub D, A, B

BLZ D, L

SLT D, A, B

BNE D, L

Variations

Condition flags

ARM

if (A < B) goto L

Section 2.16

MIPS

Sub A - B

Branch (B - A) less than 0

Sub D, A, B

BLZ D, L

SLT D, A, B

BNE D, L

Variations

Condition flags

Variations

Conditional instructions

```
top:  CMP r3, r4
      SUBGT r3, r3, r4
      SUBLT r4, r4, r3
      BNE top
```

ARM

CMP

MIPS

BLT

BGT

BEQ

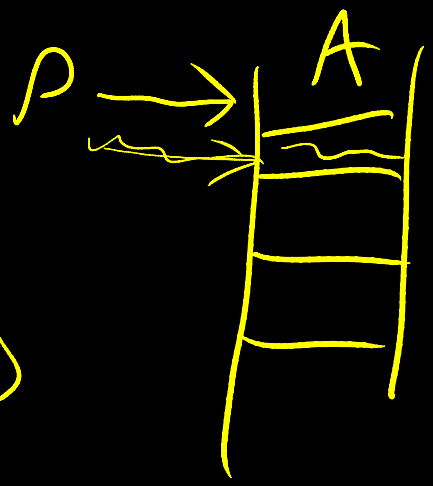
Variations

Stack operations & multiple destinations

Section 2.17

x86

1000's of inst



Push

P = P + 4

sw A, 0(P)

Pop

lw A, 0(P)

P = P - 4

Variations

String instructions

MOV AX, FFh

MOV DI, 5000h

MOV CX, 2000h

REP STOSB

Store byte reg AX into memory @DI

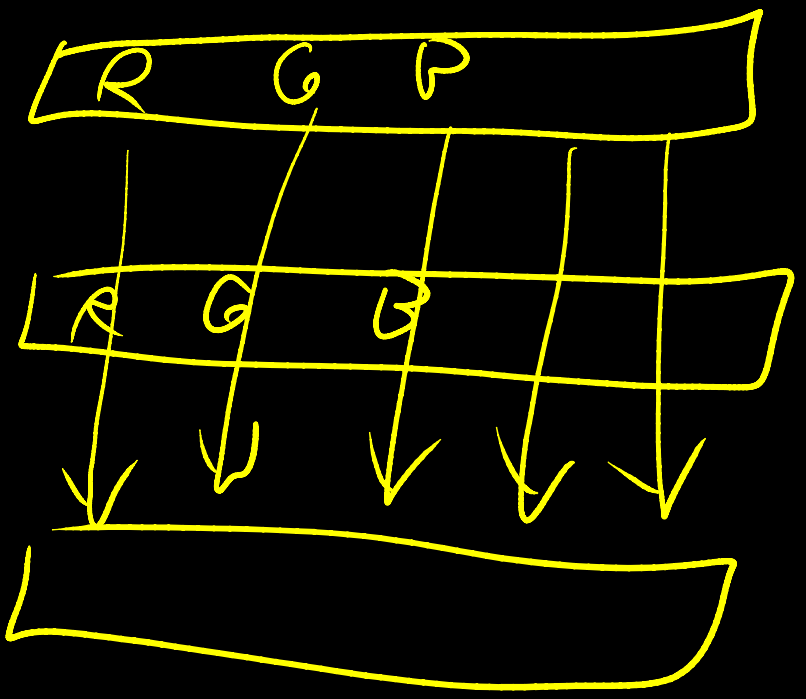
Variations

Vector instructions

vaddubs v3, v1, v2

SIMD
Single Inst
Multiple Data

Vector
add
unsigned
byte
saturated



Next time

Instruction Variations

- ARM/MIPS (RISC)
- Versus x86 (CISC)