Principles of Computer Architecture

CSE 240A

Fall 2024

Hadi Esmaeilzadeh

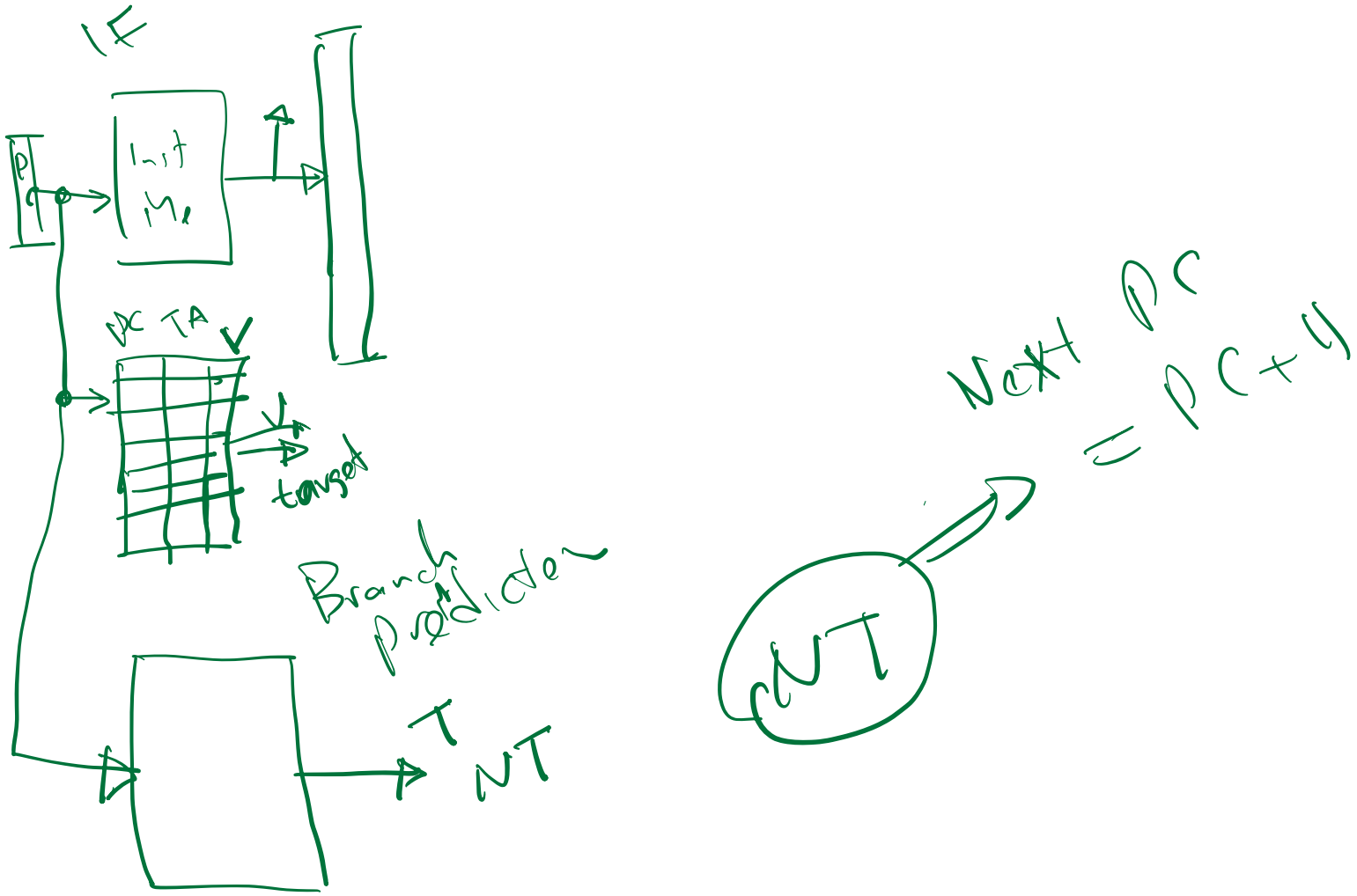
hadi@ucsd.edu

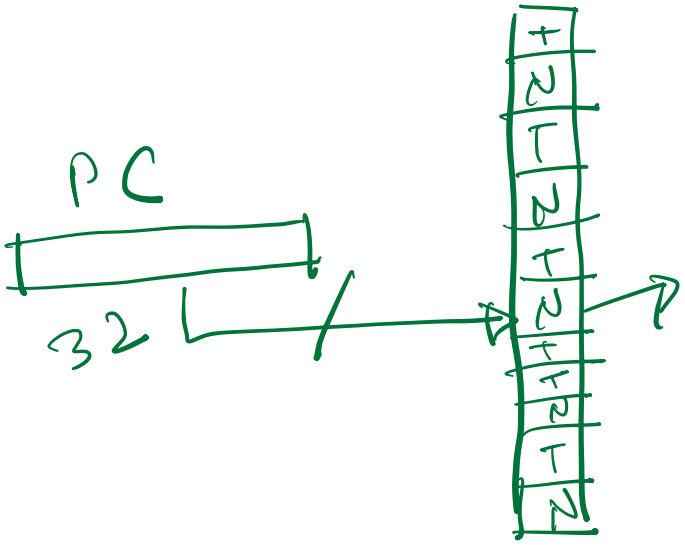
University of California, San Diego

Revisiting Branch Hazard Solutions

- Stall
- Predict Not Taken
- Predict Taken
- Branch Delay Slot

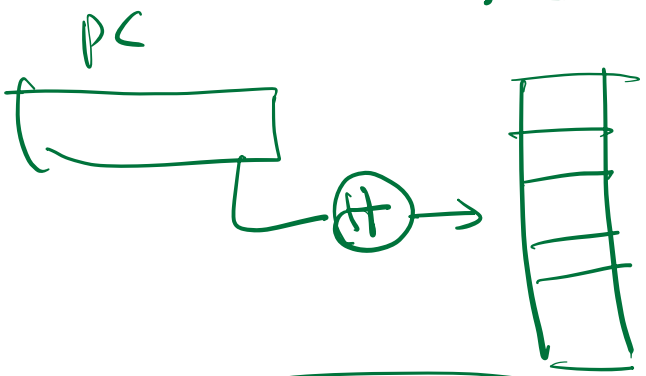
This is kind of an odds-and-ends lecture. We'll be finishing up a few things, making pipelining a little more real – dealing with some aspects that aren't quite as clean as it may have looked last lecture…


IF
Init
PC TA
V
target
Branch Predictor
T
NT
Next PC
= PC+4
NT

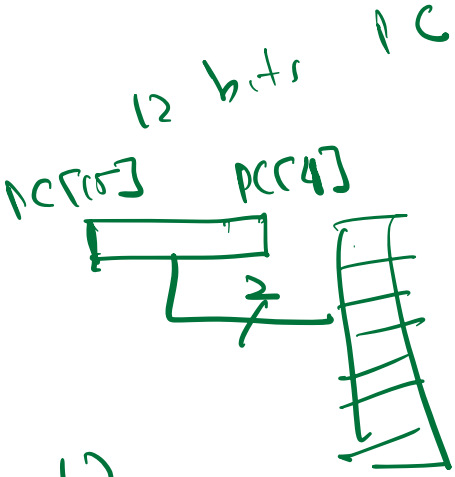


2^{32} = 4 Gig Bits

(100 MByte)

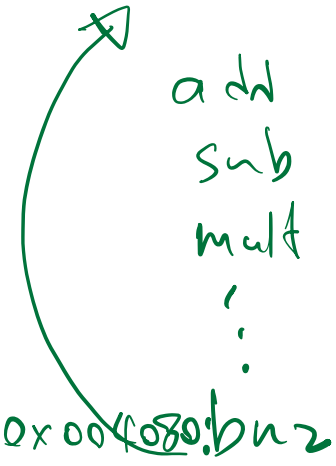


512 Byte



4 Kbits

$2^{12} = \frac{4 \times 2^{10}}{2^3}$ bit $= 2^9$



aliasing

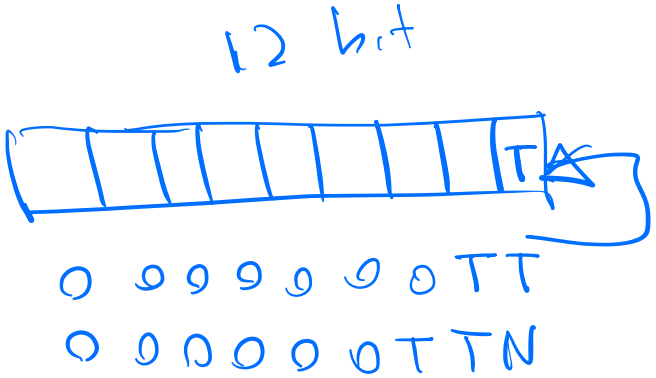
Destructive Aliasing

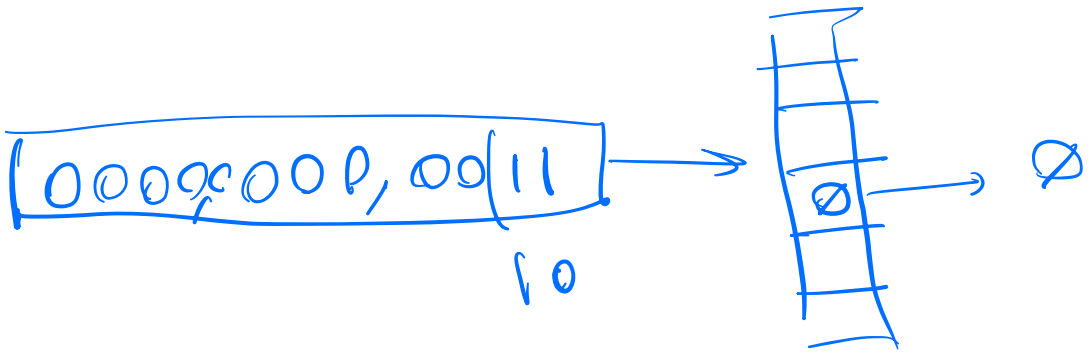
Constructive aliasing

```
if (a == 0)
    b = 1
if (b != 0)
    c = 2
if (c == 0)
    d = 1
```

Global History Register

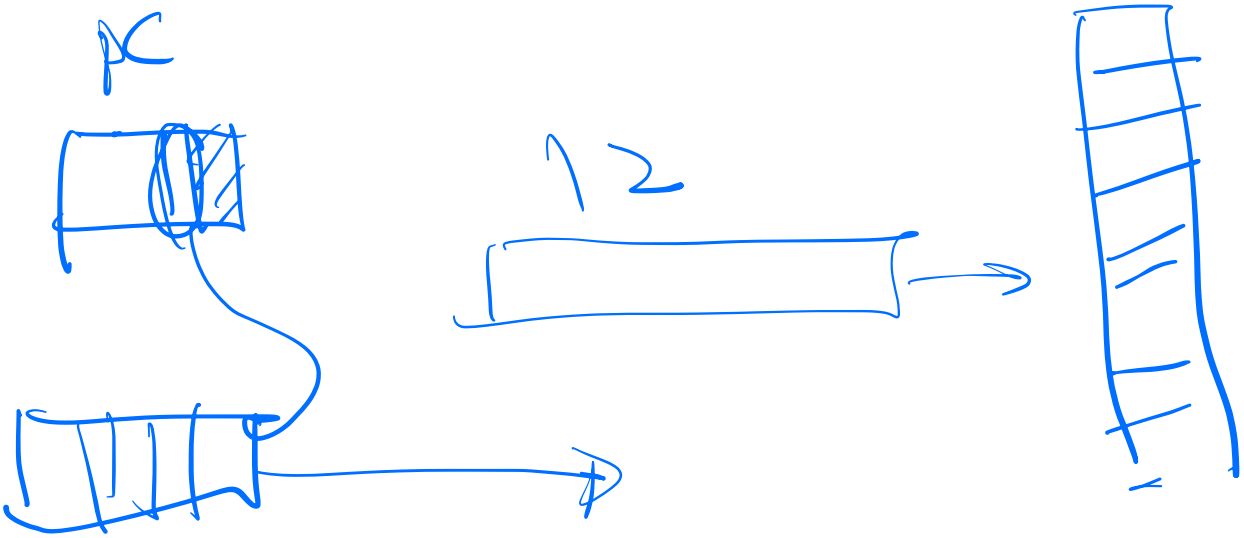
GHR

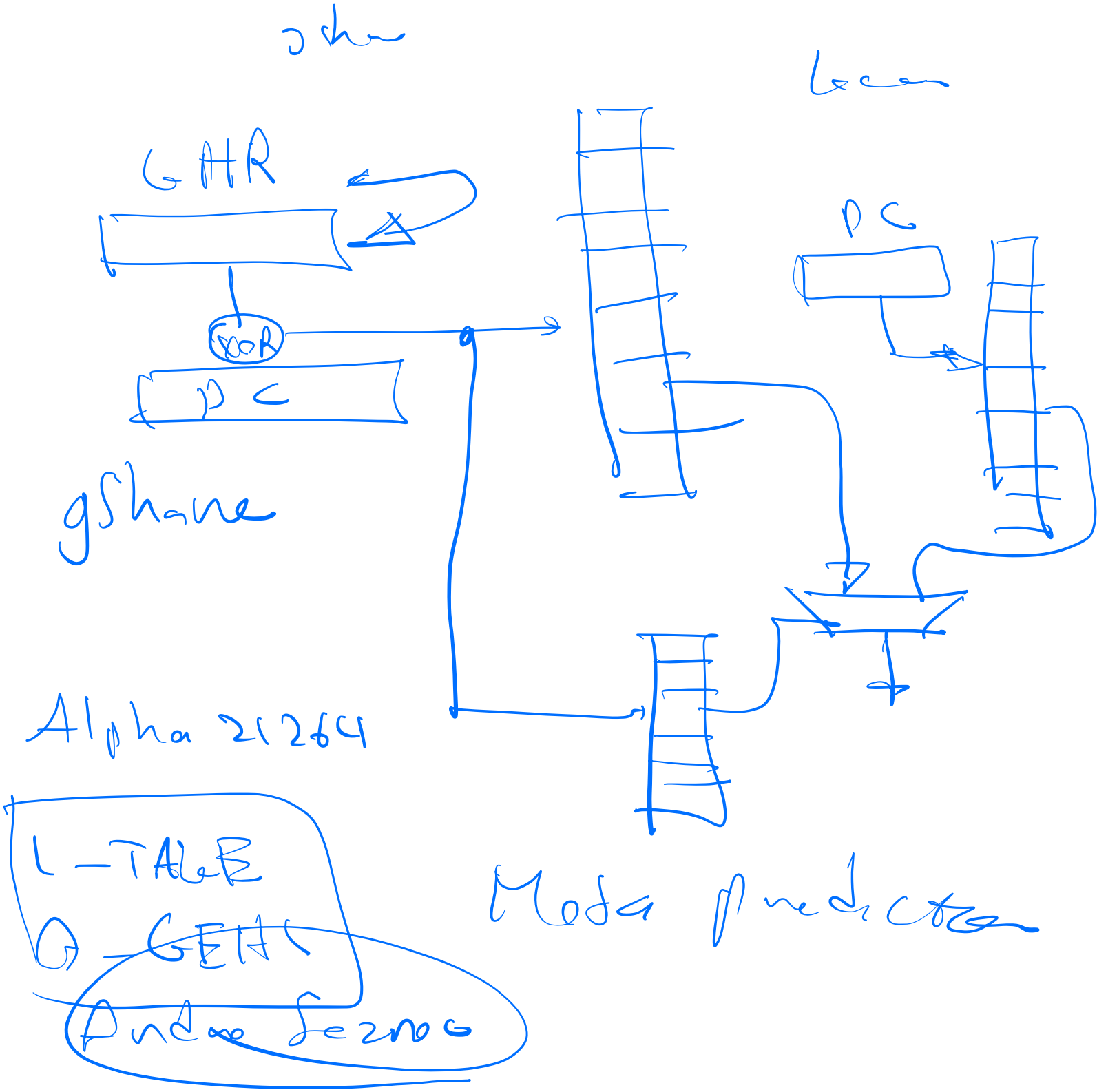


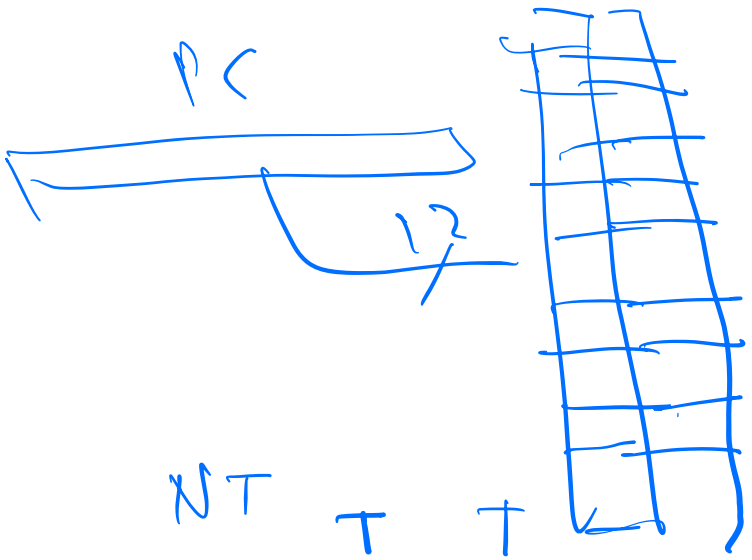


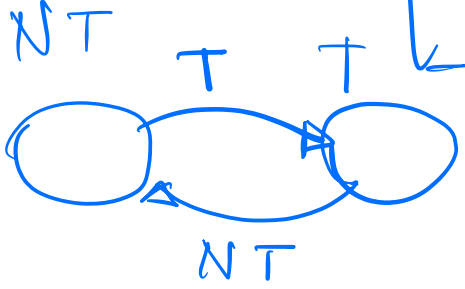
11111
111111

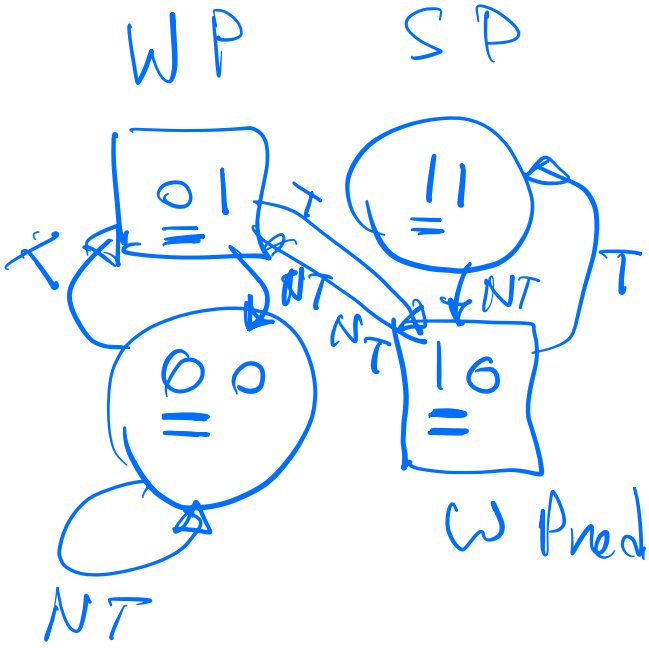
10 bit




GHR
XOR
PC
gshare
Alpha 21264
L-TAGE
O-GEHL
Andre Seznec
Meta predictor

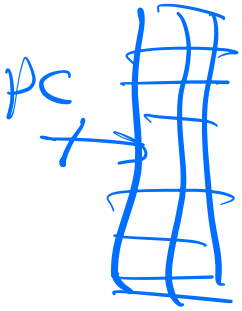


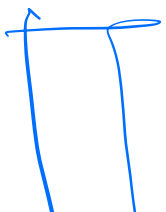




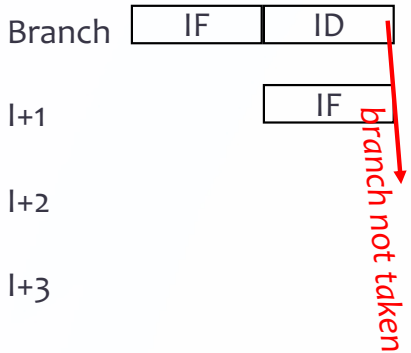
00 → 01 → 10 → 11

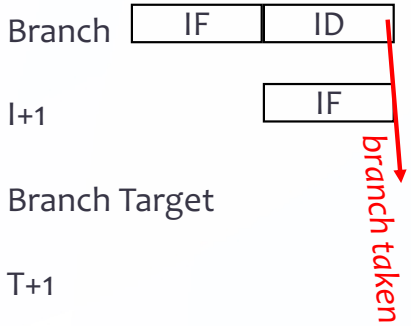
Bimodal
James Smith



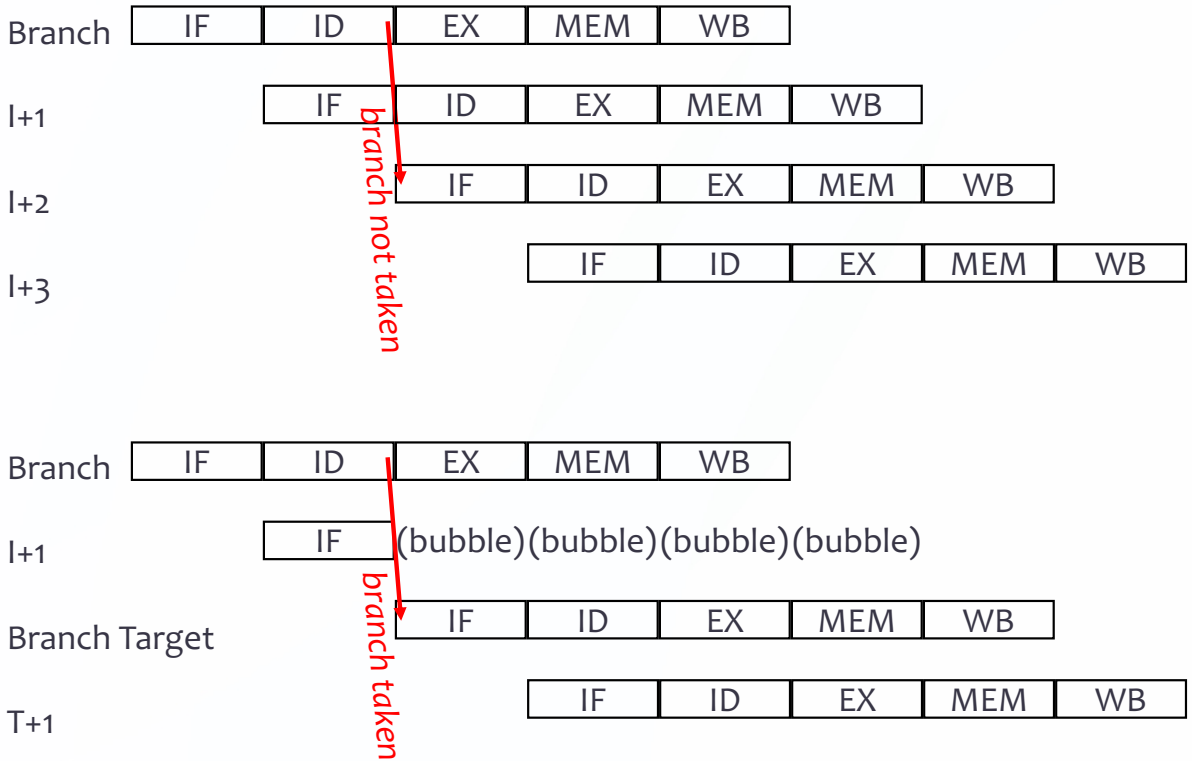


Predict Not Taken

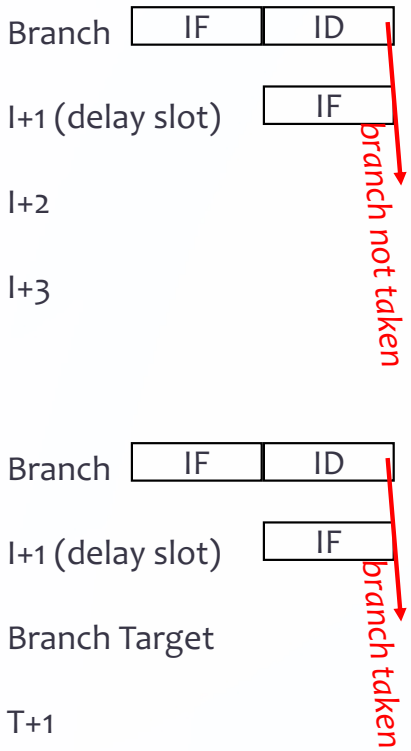




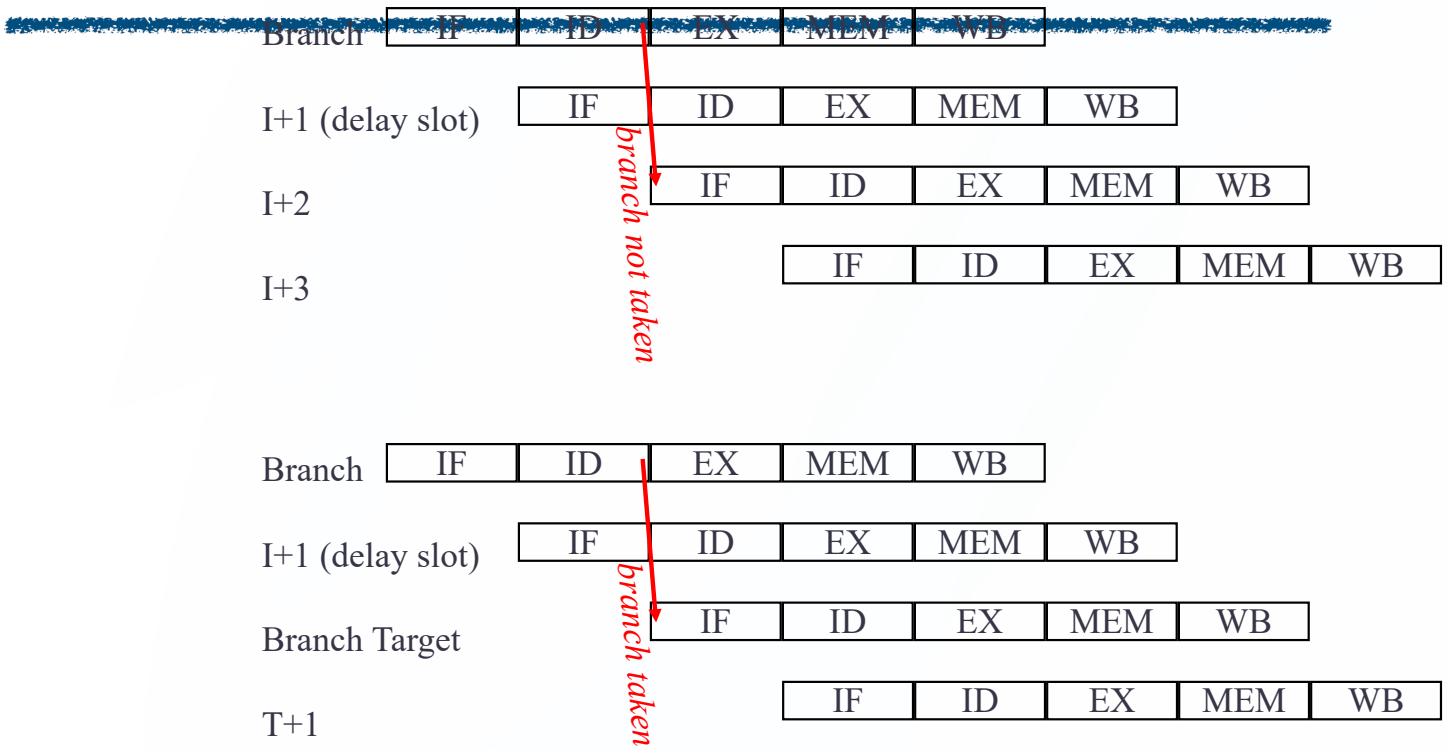
Predict Not Taken



Delayed Branch



Delayed Branch



Filling the delay slot (e.g., in the compiler)

```
lw R1, 10000(R7)
add R5, R6, R1
beqz R5, label:
[ ]
sub R8, R1, R3
add R4, R8, R9
and R2, R4, R8
```

Can be done when?
Improves performance when?

Could do sub with register rename
OR…..

```
label: add  R2, R5, R8
```

This one is okay

Problems filling delay slot

1. need to predict _____ of branch to be most effective

2. limited by _____ restriction

direction

correctness

Problems filling delay slot

1. need to predict _____ of branch to be most effective
2. limited by _____ restriction

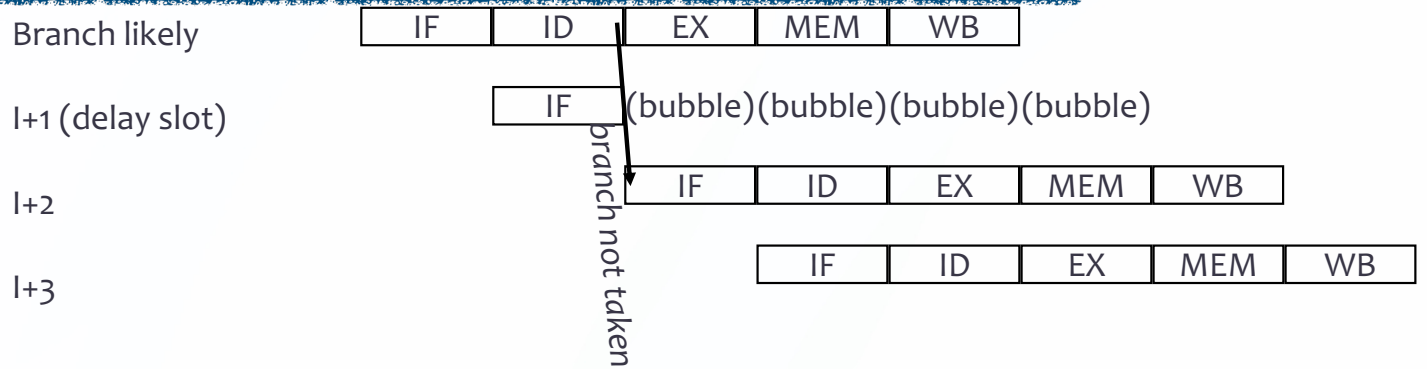
- correctness _____ restriction can be removed by a canceling branch

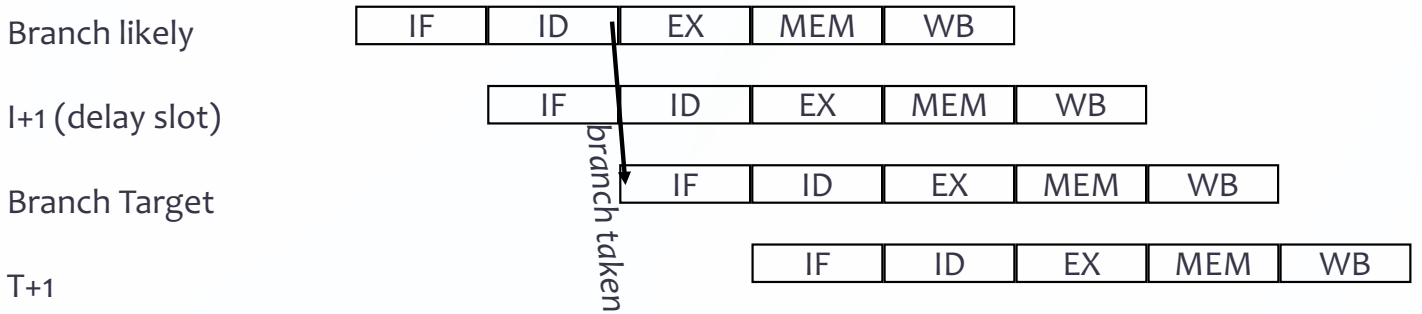
branch likely (or branch not likely???)
e.g.,

beqz likely
delay slot instruction
fall-through instruction

← squashed/nullified/canceled if branch not taken

Branch Likely





Delay Slot Utilization

- 18% of delay slots left empty
- 11% of delay slots (1) use canceling branches and (2) end up getting canceled

Data from real MIPS – 29% of Branch Delay Slot's useless. Find source…

Delay Slots, the scorecard

- Pros

Maybe cheaper for hw (no flushing)
Compiler finds useful insts
If good sked, no branch hazards

- Cons

Could be not useful dep. On sked
If cancelling, not prediction…
Breaks level of abstraction!!

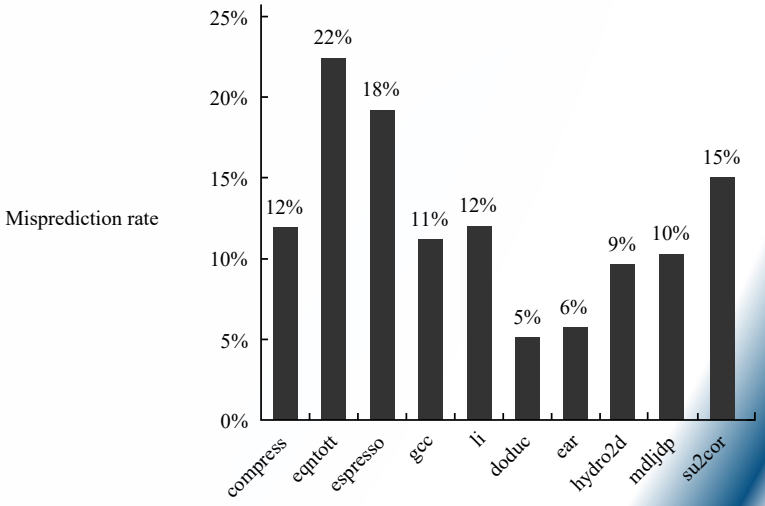
Would you include branch delay slots in your ISA if you made one today?
A. Yes B. No

Static Branch Prediction

- Static branch prediction takes place at compile time, dynamic branch prediction during program execution
- Typically requires some ISA support (eg branch_likely)
- static bp done by software, dynamic bp done in hardware
- How to make static branch predictions?
- Static branch prediction enables
  - more effective code scheduling around hazards (how?)
  - more effective use of delay slots

Schedule for more likely hazards.

Spec92, profile-driven static prediction.



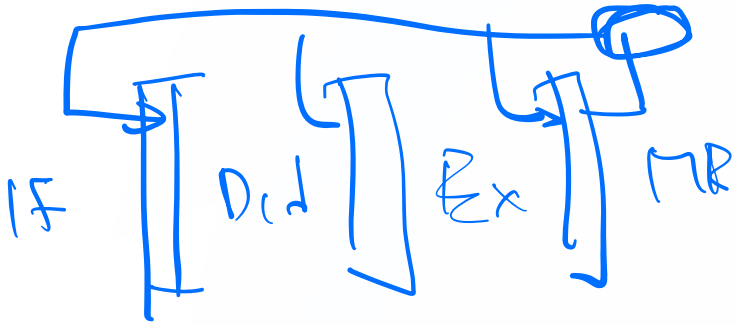
Static Branch Prediction

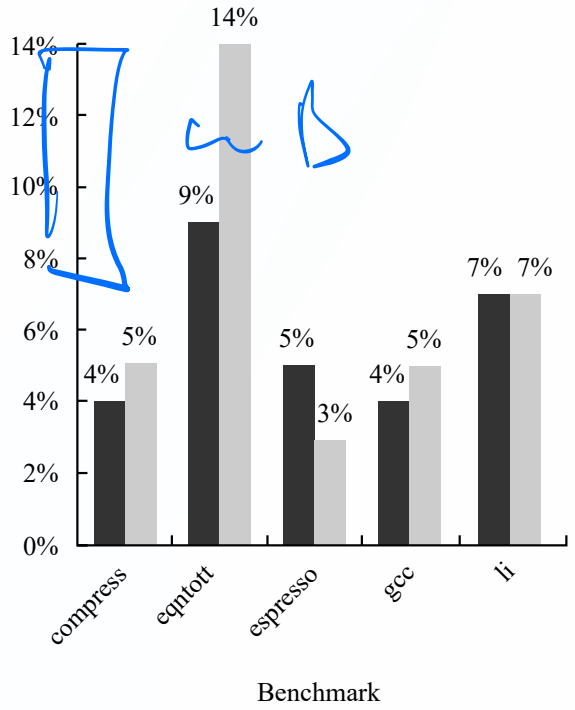
- What's might be an obstacle to doing static branch prediction (i.e., having the compiler denote which branches are biased taken/not-taken)?

Profiling (can you trust?)
Branch has diff behavior!
Breaks abstraction:
- Compiler knows issue
- Opcode implicit message
- Change ISA for this

MIPS 5-stage Integer Pipeline Performance

- Only stalls for load hazards and branch hazards, both of which can be reduced (but not eliminated) by software





segue

But now, the real world interrupts…

- Pipelining is not as easy as we have made it seem so far…
  - interrupts and exceptions
  - long-latency instructions

Exceptions and Interrupts

- Transfer of control flow (to an exception handler) without an explicit branch or jump
- are often unpredictable
- examples
  - I/O device request
  - OS system call
  - arithmetic overflow/underflow
  - FP error
  - page fault
  - memory-protection violation
  - hardware error
  - undefined instruction

Which of the following events are synchronous (consistently occur at the same time in a program given the same input and same memory allocation)

1. ALU overflow exception
2. unconditional branch
3. timer interrupt
4. I/O interrupt

Selection	Synchronous
A	1, 3, 4
B	1, 2
C	2,3
D	3,4
E	None of the above

Classes of Exceptions

- synchronous vs. asynchronous
- user-initiated vs. coerced
- user maskable vs. nonmaskable
- within instruction vs. between instructions
- resume vs. terminate

Diff based on OS
(Doom vs. space shuttle)

Basic Exception Methodology

- What do you need to do?

Save the reason, change PC. For the OS, save program state

Basic Exception Methodology

- turn off writes for faulting instruction and following
- force a trap into the kernel at the next IF
- save the cause in a register OR go to a specific place in the kernel based on the cause
- save the PC of the faulting instruction

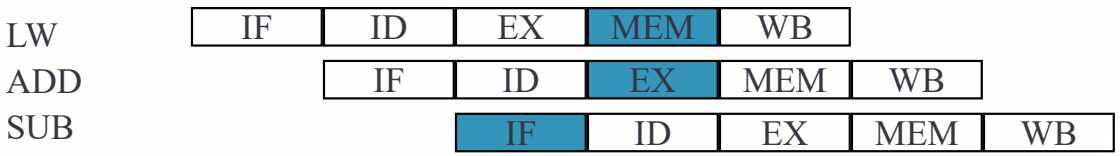
When an exception occurs, for which type of instruction is saving only the current pc insufficient for the OS to resume execution of the program later.

A. Branch
B. Jump and link
C. Delayed branch
D. Load Word
E. More than one of the above

Exceptions Can Occur In Several Places in the pipeline

- IF -- page fault on memory access, misaligned memory access, memory-protection violation
- ID -- illegal opcode
- EX -- arithmetic exception
- MEM -- page fault, misaligned access, memory-protection
- WB -- none

(and, of course, asynchronous can happen anytime)



Simplifying Exceptions in the ISA

1. Each instruction changes machine state only once
    1. autoincrement
    2. string operations
    3. condition codes
2. Each instruction changes machine state at the end of the pipeline (when you know it will not cause an exception)

Enables precise interrupts
string ops are rough?
segue

But now, the real world interrupts…

- Pipelining is not as easy as we have made it seem so far…
  - interrupts and exceptions
  - long-latency instructions

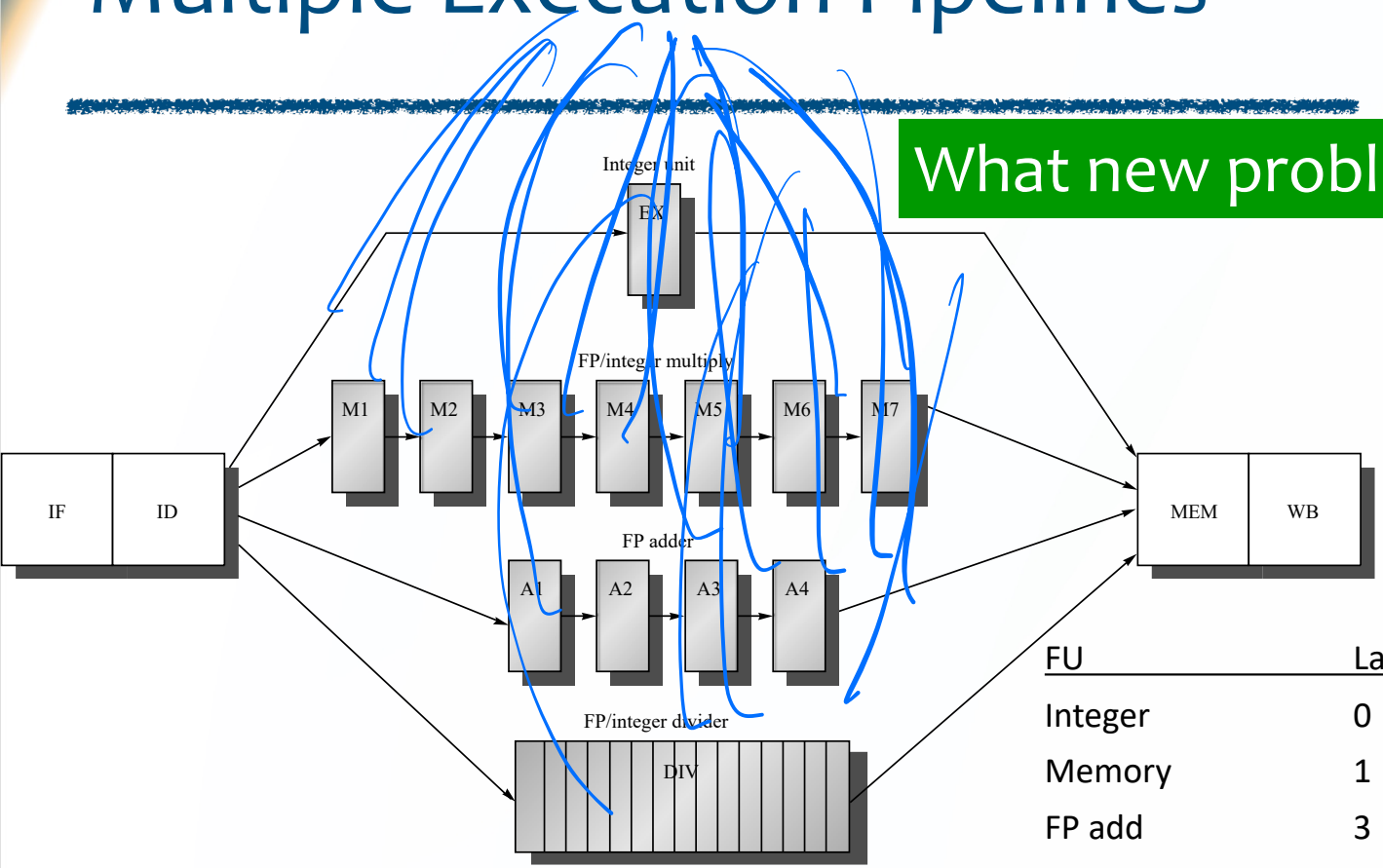
Handling Multicycle Operations

- Unrealistic to expect that all operations take the same amount of time to execute
- ____, some _____will take longer
- This violates some of the assumptions of our simple pipeline

fp

Memory ops

Multiple Execution Pipelines



What new problems are introduced?

FU	Latency	Initiation interval
Integer	0	1
Memory	1	1
FP add	3	1
FP multiply	6	1
FP divide	24	24

Which problems are introduced by variable length pipelines?

1. Structural hazards
2. WAR Hazards
3. WAW Hazards
4. Out of order completion

Selection	Now a problem
A	1, 2, 3
B	1, 2, 4
C	2, 3
D	1, 3, 4
E	None of the above

New problems

- structural hazards are ___Yes_____
  - divide unit
  - WB stage
- WAW hazards are __Yes___
- out-of-order completion is ___Yes____
- WAR hazards are _____No_____

Yes to all except WAR (not yet)

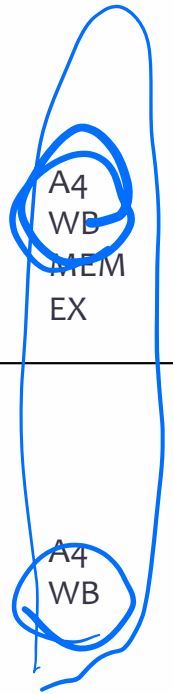
Structural hazards and WAW hazards

- structural hazards
  - divide unit
  - WB stage

ADDD	IF	ID	A1	A2	A3	A4	MEM	WB
…		IF	ID	EX	MEM	WB		
…			IF	ID	EX	MEM	WB	
LD				IF	ID	EX	MEM	WB

- WAW hazards

ADDD F8, …	IF	ID	A1	A2	A3	A4	MEM	WB
LD F8, …		IF	ID	EX	MEM	WB		



Which of the following becomes much more complicated if we have out-of-order completion?

Selection	Now a problem
A	Branches
B	Exceptions
C	Both
D	Neither

New problems

- structural hazards
  - divide unit
  - WB stage
- WAW hazards are possible
- out-of-order completion
- WAR hazards still not possible

New problems

- structural hazards
  - divide unit
  - WB stage
- WAW hazards are possible
- out-of-order completion
- WAR hazards still not possible

- (We'll ignore solutions to these problems, for now, because our architecture for out-of-order execution will subsume those solutions)

Key Points

- Data Hazards can be significantly reduced by forwarding
- Branch hazards can be reduced by early computation of condition and target, branch delay slots, branch prediction
- Data hazard and branch hazard reduction require complex compiler support
- Exceptions are hard
- variable-length instructions introduce structural hazards, WAW hazards, more RAW hazards