6.004 Tutorial Problems
L19 – Data Hazards in Pipelined Processors

Resolving Data Hazards by Stalling

- Strategy 1: Stall. Wait for the result to be available by freezing earlier pipeline stages

```
addi x11, x10, 2
xor x13, x11, x12
sub x17, x15, x16
xori x19, x18, 0xF
```

Stall

	1	2	3	4	5	6	7	8
IF	addi	xor	sub	*sub*	*sub*	*sub*	xori	
DEC		addi	xor	*xor*	*xor*	*xor*	sub	xori
EXE			addi	**NOP**	**NOP**	**NOP**	xor	sub
MEM				addi	**NOP**	**NOP**	**NOP**	xor
WB					addi	**NOP**	**NOP**	**NOP**

x11 updated

Stalls increase CPI!

Resolving Data Hazards by Bypassing

- Strategy 2: Bypass. Route data to the earlier pipeline stage as soon as it is calculated

```
addi x11, x10, 2
xor x13, x11, x12
sub x17, x15, x16
xori x19, x18, 0xF
```

- addi writes to x11 at the end of cycle 5… but the result is produced during cycle 3, at the EXE stage!

	1	2	3	4	5
IF	addi	xor	sub	xori	
DEC		addi	xor	sub	xori
EXE			addi	xor	sub
MEM				addi	xor
WB					addi

addi result computed

x11 updated

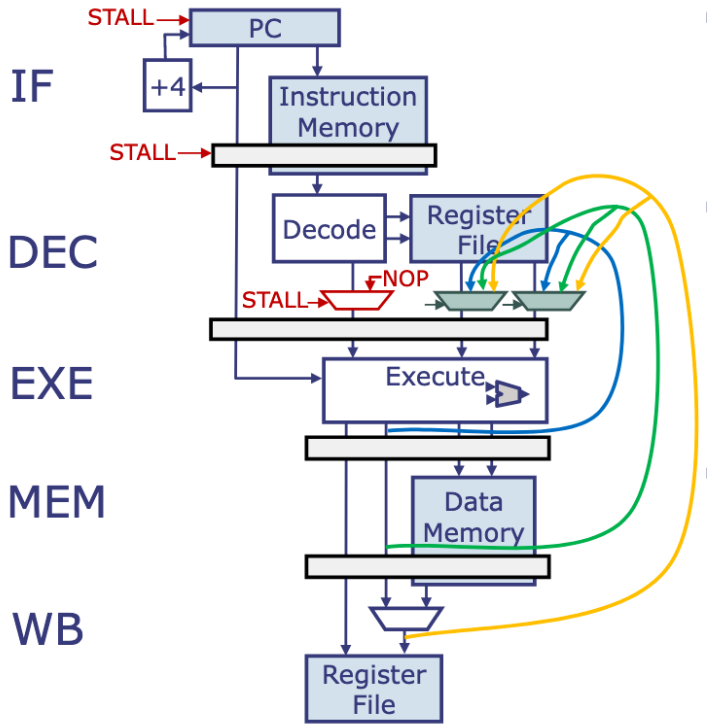
Load-To-Use Stalls

- Bypassing cannot eliminate load delays because their data is not available until the WB stage
- Bypassing from WB still saves a cycle:

```
lw x11, 0(x10)
xor x13, x11, x12
sub x17, x15, x16
xori x19, x18, 0xF
```

	1	2	3	4	5	6	7	8
IF	lw	xor	sub	*sub*	*sub*	xori		
DEC		lw	xor	*xor*	*xor*	sub	xori	
EXE			lw	**NOP**	**NOP**	xor	sub	xori
MEM				lw	**NOP**	**NOP**	xor	sub
WB					lw	**NOP**	**NOP**	xor

lw data available ↑ ↖ x11 updated



Problem 1. ★

The program shown on the right is executed on a 5-stage pipelined RISC-V processor with full bypassing.

The program has been running for a while and execution is halted at the end of cycle 105.

The pipeline diagram shown below shows the history of execution at the time the program was halted.

```
  . = 0
outer_loop:
  addi x11, x0, 16   // initialize loop index J
  addi x12, x0, 0

loop:                // add up elements in array
  addi x11, x11, -1  // decrement index
  slli x13, x11, 2   // convert to byte offset
  lw x14, 0x310(x13) // load value from A[J]
  add x12, x12, x14  // add to sum
  bnez x11, loop

  j outer_loop       // perform test again!
```

(A) Please indicate on which cycle(s), 100 through 105, each of the following actions occurred. If the action did not occur in any cycle, write "NONE".

cycle	*100*	*101*	*102*	*103*	*104*	*105*
IF	slli	lw	add	bnez	bnez	bnez
DEC	addi	slli	lw	add	add	add
EXE	NOP	addi	slli	lw	NOP	NOP
MEM	NOP	NOP	addi	slli	lw	NOP
WB	bnez	NOP	NOP	addi	slli	lw

Register value used from Register File: _____

Register value bypassed from EXE stage to DEC stage: _____

Register value bypassed from MEM stage to DEC stage: _____

Register value bypassed from WB stage to DEC stage: _____

(B) Why is the NOP instruction inserted in cycle 104?

Problem 2. ★

The following program fragments are being executed on the 5-stage pipelined RISC-V processor with full bypassing. For each fragment, the pipeline diagram shows the state of the pipeline for cycle 1000 of execution. Please fill in the diagram for cycle 1001; use "?" if you cannot tell what opcode to write into a stage. Then for **both** cycles use arrows to indicate any bypassing from the EXE/MEM/WB stages back to the DEC stage.

(A)

```
…
sw x1,  0(x0)
lw x17, 0xC(x1)
addi x2, x2, -4
slli x11, x17, 2
sw x11, 0(x2)
jal ra, fact
…
```

Cycle	*1000*	*1001*
IF	`sw`	
DEC	`slli`	
EXE	`addi`	
MEM	`lw`	
WB	`sw`	

(B)

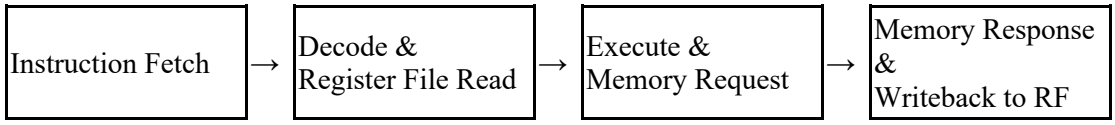
```
…
xor x11, x11, x12
slli x12, x12, 3
sub x13, x12, x11
and x12, x13, x11
add x13, x12, x13
sw x13, 0x100(x0)
…
```

Cycle	*1000*	*1001*
IF	`add`	
DEC	`and`	
EXE	`sub`	
MEM	`slli`	
WB	`xor`	

Problem 3.

For the following questions, assume that you are running code on the 4-stage pipelined processor whose stages are shown below. Also assume that:

- ⇨ there is no bypassing,
- ⇨ the register file (RF) is *not* a bypass register file,
- ⇨ there are no early writebacks (meaning that every instruction must go through the last stage),
- ⇨ there is a register between pipeline stages (not a FIFO buffer), and
- ⇨ data memory responds immediately (in the next clock cycle) to any load request.



The problems will ask about pipeline hazards; we will say a hazard is ***observable*** if it causes the pipeline to stall.

Note: For some problems it might be helpful to draw a pipeline diagram. There are several blank pipeline diagrams on the next page that you can use as well as additional ones on the last page of the exam.

```
0x100       addi sp, sp, -4
0x104       srli a1, a0,  1
0x108       addi a4, a4, +1
0x10C       beqz a0, L0
0x110       andi a2, a1, 0x001
0x114   L0: sub  a3, a0, a2
0x118       lw   a5, +4(sp)
0x11C       sw   a3,  0(sp)
```

(A) (3 points) Identify all the potential read-after-write (RAW) data hazards in the code above (including ones that are not observable). The number at the beginning of each line corresponds to the address of that line of code. For each hazard, write **Lines *a* and *b*** if a register is written in the line at address ***a*** and read in the line at address ***b***. If there are more spaces than you need, leave the extra spaces blank.

(1) Lines _____ and _____

(2) Lines _____ and _____

(3) Lines _____ and _____

(4) Lines _____ and _____

(5) Lines _____ and _____

(6) Lines _____ and _____

For each of the scenarios below, whenever you are asked to list the observable hazards, specify the number to the left of the hazard you are referring to from part A. So, if the hazards labelled (1) and (3) in part A of your answer are observable, the enter 1, 3 as your observable hazards response. Enter None if there are no observable hazards.

(B) (4 points) Which of the hazards from part A are observable when the branch (address 0x10C***) is taken*** and is correctly predicted as being taken?

List of observable hazards or None: _____

(C) (4 points) Which of the hazards from part A are observable when the branch is ***not taken*** and is correctly predicted as being not taken? How many nops are inserted (i.e., for how many cycles is the pipeline stalled) in this case?

List of observable hazards or None: _____

Number of nops/stalled cycles:_____

(D) (4 points) Now reconsider part (C) where the branch is ***not taken*** and is correctly predicted as being not taken. However, this time assume that there is a cache miss fetching the `lw` instruction at address 0x118 causing a three cycle stall. Which hazards are now observable?

List of observable hazards or None: _____

(E) (5 points) Once again reconsider part (C) where the branch is ***not taken*** and is correctly predicted as being not taken. However, now suppose we add bypassing from the writeback stage to the decode stage (or equivalently, we replace the register file with a bypass register file), and the instruction cache no longer misses. Now which hazards are observable? How many nops are inserted (i.e., for how many cycles is the pipeline stalled) in this case?

List of observable hazards or None: _____

Number of nops/stalled cycles:_____

IF									
DEC									
EXE									
WB									

IF									
DEC									
EXE									
WB									

IF									
DEC									
EXE									
WB									