

6.004 Tutorial Problems
L19 – Control Hazards in Pipelined Processors

Problem 1. ★

The loop on the right has been executing for a while on our standard 5-stage pipelined RISC-V processor with branch annulment and full bypassing.

```

...
L1: addi x10, x10, -4
    slti x11, x10, 10
    beqz x11, L2
    lw x12, 0x200(x10)
    j L3
L2: lw x12, 0x300(x10)
L3: sw x12, 0x400(x0)
    bnez x10, L1
    addi x12, x12, 1
    xor x12, x12, x0
...

```

Cycle #	300	301	302	303	304	305	306	307	308	309
IF	addi	slti	beqz	lw	j	lw	sw	bnez	bnez	bnez
DEC	NOP	addi	slti	beqz	lw	NOP	lw	sw	sw	sw
EXE	NOP		addi	slti	beqz	NOP	NOP	lw	NOP	NOP
MEM	bnez			addi	slti	beqz	NOP	NOP	lw	NOP
WB	sw				addi	slti	beqz	NOP	NOP	lw

- (A) Fill in the pipeline diagram for cycles 300-309 assuming that at cycle 300 the instruction at L1 is fetched. Also, assume that the branch to L2 is taken, as well as the final branch back to L1. Indicate which bypass/forwarding paths are active in each cycle by drawing a vertical arrow in the pipeline diagram from pipeline stage X in a column to the RF stage in the same column if an operand would be bypassed from stage X back to the RF stage that cycle. Note that there may be more than one vertical arrow in a column.

Fill in pipeline diagram including bypass arrows in pipeline diagram above

- (B) Assume that the previous iteration of the loop executed the same instructions as the iteration shown here. Please complete the pipeline diagram for cycle 300 by filling in the OPCODEs for the instructions in the DEC, EXE, MEM, and WB stages.

Fill in OPCODEs for Cycle 300

Cycle #	300	...	305	306	307	308	309	310	311	312
IF	addi		lw	sw	bnez	bnez	bnez	addi	xor	addi
DEC	NOP		NOP	lw	sw	sw	sw	bnez	addi	NOP
EXE	NOP		NOP	NOP	lw	NOP	NOP	sw	bnez	NOP
MEM	bnez		beqz	NOP	NOP	lw	NOP	NOP	sw	bnez
WB	sw		slti	beqz	NOP	NOP	lw	NOP	NOP	sw

(C) Indicate which branches are taken by providing the cycle in which the taken branch instruction enters the IF stage.

Cycle number(s) or NONE: 302, 307

(D) During which cycle(s), if any, do we have stalled instructions?

Cycle number(s) or NONE: 307, 308

Now consider a modified processor, P2, which has extra hardware in the decode stage (DEC) to resolve simple branches one cycle earlier: the decode stage includes both a circuit to check whether a register is equal to zero, and an extra adder to compute the branch target for taken branches. This processor can thus compute nextPC for beqz and bnez in DEC instead of EXE.

(E) Redo part A using processor P2 assuming the same path is taken through the code.

Cycle #	300	301	302	303	304	305	306	307	308	309
IF	addi	slti	beqz	lw	lw	sw	bnez	bnez	bnez	addi
DEC	NOP	addi	slti	beqz	NOP	lw	sw	sw	sw	bnez
EXE	bnez		addi	slti	beqz	NOP	lw	NOP	NOP	sw
MEM	sw			addi	slti	beqz	NOP	lw	NOP	NOP
WB	NOP				addi	slti	beqz	NOP	lw	NOP

(F) Compare the number of cycles per loop iteration using the original processor and the modified processor.

Cycles per loop in original processor: 12

Cycles per loop in processor P2: 10

Problem 2. ★

You've discovered a secret room in the basement of the Stata center full of discarded 5-stage pipelined RISC-V processors. Unfortunately, many have certain defects. You discover that they fall into four categories:

- C1:** A modified, completely functional 5-stage RISC-V processor with working bypass paths, annulment, and other components, as well as extra hardware support that allows the nextPC to be calculated in the Decode stage.
- C2:** A defective version of C1 with a bad register file: all data read from the register file is zero.
- C3:** A defective version of C1 with broken bypass muxes: all source operands come from the register file, even if they should be read from bypassed paths.
- C4:** A defective version of C1 without annulment of instructions following branches.

To help sort the processors into the above classes, you write the following small test program:

<pre> . = 0x0 // Start at 0x0, with ZERO // in all registers... addi x10, x0, 4 jal x12, X slli x12, x12, 1 x: addi x12, x12, -4 add x13, x12, x10 jr x13 </pre>	C1 x10 = 4 x12 = 8 ----- x12 = 4 x13 = 8	C2 x10 = 4 x12 = 8 ----- x12 = 4 x13 = 4	C3 x10 = 4 x12 = 8 ----- x12 = -4 x13 = 4	C4 x10 = 4 x12 = 8 ----- x12 = 16 x12 = 12 x12 = 16
---	--	--	---	--

want x12 from bypass
 want x10 from RF

Your plan is to single-step through the program using each processor, carefully noting the address the final `jr` loads into the PC. Your goal is to determine which of the above four classes a chip falls into by this `jr` address.

For each class of RISC-V processor described above, specify the value that will be loaded into the PC by the final `jr` instruction.

Pipeline diagram showing first 7 cycles of test program executing on C1:

<i>cycle</i>	0	1	2	3	4	5	6
IF	addi	jal	slli	addi	add	jr	
DEC		addi	jal	NOP	addi	add	jr
EXE			addi	jal	NOP	addi	add
MEM				addi	jal	NOP	addi
WB					addi	jal	NOP

C1: jr goes to address: 8

C2: jr goes to address: 4

C3: jr goes to address: 0

C4: jr goes to address: 16

Problem 3. ★

(A) How many cycles does it take to run each iteration of the following loop on a standard 5-stage pipelined RISC-V processor?

```
loop: lw x10, 0x100(x0)
      beqz x10, loop
      add x12, x10, x11
      sub x13, x12, x1
```

Number of cycles per loop iteration: 6

<i>cycle</i>	0	1	2	3	4	5	6	7	8	9	10
IF	lw	beqz	add	add	add	sub	lw				
DEC		lw	beqz	beqz	beqz	add	NOP	lw			
EXE			lw	NOP	NOP	beqz	NOP	NOP	lw		
MEM				lw	NOP	NOP	beqz	NOP	NOP	lw	
WB					lw	NOP	NOP	beqz	NOP	NOP	lw

6 cycles

(B) Assuming a defective 5-stage pipelined RISC-V processor where the instructions following a taken branch are not annulled, which of the following statements would be true?

1. The add instruction would be executed each time through the loop.
2. The loop would take 5 cycles to execute
3. The value of the register x10 that is tested by the beqz instruction comes from a bypass path.
4. The value of register x10 that is accessed by the add instruction comes from the register file.

For such a defective processor, the instruction flow diagram looks like this:


<i>cycle</i>	0	1	2	3	4	5	6	7	8	9	10
IF	lw	beqz	add	add	add	sub	lw				
DEC		lw	beqz	beqz	beqz	add	sub	lw			
EXE			lw	NOP	NOP	beqz	add	sub	lw		
MEM				lw	NOP	NOP	beqz	add	sub	lw	
WB					lw	NOP	NOP	beqz	add	add	lw

6 cycles

(C) Consider a modified processor, P2, which has extra hardware for the special case of checking if a register is equal to zero or not in the decode stage. What would be the number of cycles per loop iteration in this case?

Number of cycles per loop iteration on processor P2: 5

<i>cycle</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>
IF	lw	beqz	add	add	add	lw					
DEC		lw	beqz	beqz	beqz	NOP	lw				
EXE			lw	NOP	NOP	beqz	NOP	lw			
MEM				lw	NOP	NOP	beqz	NOP	lw		
WB					lw	NOP	NOP	beqz	NOP	lw	

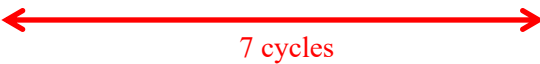


5 cycles

(D) Now consider a third processor, P3, whose instruction and data memories are pipelined and take 2 clock cycles to respond. Assume that P3 also has the extra hardware for checking if a register is equal to zero or not in the decode stage. What would be the number of cycles per loop iteration using P3?

NOP Number of cycles per loop iteration on processor P3: 7

<i>cycle</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>	<i>10</i>	<i>11</i>	<i>12</i>	<i>13</i>
IF	lw	beqz	add	sub	sub	sub	sub	lw						
		lw	beqz	add	add	add	add	NOP	lw					
DEC			lw	beqz	beqz	beqz	beqz	NOP	NOP	lw				
EXE				lw	NOP	NOP	NOP	beqz	NOP	NOP	lw			
MEM					lw	NOP	NOP	NOP	beqz	NOP	NOP	lw		
						lw	NOP	NOP	NOP	beqz	NOP	NOP	lw	
WB							lw	NOP	NOP	NOP	beqz	NOP	NOP	lw



7 cycles

Problem 4.

You've been given a 5-stage pipelined RISC-V processor. Unfortunately, the processor you've been given is defective: it has no bypass paths, annulment of instructions in branch delay slots, or pipeline stalls.

```

        nop
        nop
        nop
        nop

loop:
    lw x10, 0x0(x10)
AA:
    sll x14, x10, x11
BB:
    bnez x10, loop
CC:
    add x13, x10, x13

    nop
    nop
    nop
    nop
```

You undertake to convert some existing code, designed to run on an unpipelined RISC-V, to run on your defective pipelined processor. The scrap of code on above is a sample of the program to be converted. It doesn't make much sense to you – it doesn't to us either – but you are to add the **minimum** number of **NOP** instructions at the various tagged points in this code to make it give the same results on your defective pipelined RISC-V as it gives on a normal, unpipelined RISC-V.

Note that the code scrap begins and ends with sequences of **NOPs**; thus, you don't need to worry about pipeline hazards involving interactions with instructions outside of the region shown.

(A) Specify the minimal number of **NOP** instructions (*defined as `add x0, x0, x0`*) to be added at each of the labeled points in the above program.

NOPs at Loop: 0

NOPs at AA: 3

NOPs at BB: 0

NOPs at CC: 2

Below is a diagram of the instruction flow for this program. Note that nextPC is available in the execute stage of the bnez instruction even without bypassing.

<i>cycle</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>
IF	lw	NOP	NOP	NOP	sll	bnez	NOP	NOP	lw/add
DEC		lw	NOP	NOP	NOP	sll	bnez	NOP	NOP
EXE			lw	NOP	NOP	NOP	sll	bnez	NOP
MEM				lw	NOP	NOP	NOP	sll	bnez
WB					lw	NOP	NOP	NOP	sll

(B) On a **fully functional** 5-stage pipeline (with working bypass, annul, and stall logic), the above code will run fine with no added **NOPs**. How many clock cycles of execution time are required by the fully functional 5-stage pipelined RISC-V **for each iteration** through the loop?

Clocks per loop iteration: 7

Below is a diagram of the instruction flow for this program.

<i>cycle</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>
IF	lw	sll	bnez	bnez	bnez	add	NOP	lw
DEC		lw	sll	sll	sll	bnez	add	NOP
EXE			lw	NOP	NOP	sll	bnez	NOP
MEM				lw	NOP	NOP	NOP	bnez
WB					lw	NOP	NOP	NOP