ENCM 369 Winter 2023 Lab 9 for the Week of March 20

Steve Norman
Department of Electrical & Software Engineering
University of Calgary

March 2023

Administrative details

You may work with a partner on this assignment

If you choose to work with a partner, please make sure that both partners fully understand all parts of your assignment submission, and please follow the instructions regarding submission of your PDF document.

Partners must be in the same lab section. The reason for this rule to keep teaching assistant workloads balanced and to make it as easy as possible for teaching assistants to record marks.

Due Dates

As with Lab 8, the due dates don't follow the usual pattern (but they do follow the Lab 8 pattern).

The Due Date for this assignment is 6:00pm Thursday, March 30.
The Late Due Date is 6:00pm Friday, March 31

The penalty for handing in an assignment after the Due Date but before the Late Due Date is 3 marks. In other words, X/Y becomes (X–3)/Y if the assignment is late. There will be no credit for assignments turned in after the Late Due Date; they will not be marked.

How to package and hand in your assignments

You must submit your work as a *single PDF file* to the D2L dropbox that will be set up for this assignment. The dropbox will be configured to accept only file per student, but you may replace that file as many times as you like before the due date.

See the Lab 1 instructions for more information about preparing and uploading your PDF file.

Important update for those working with a partner: Please submit only one PDF file for both students. On the cover page, put lab section, name and ICID information in this format:

Group Submission for [lab section]

Submitted by: [submitter's name]
UCID: [submitter's UCID]

Partner: [partner's name]
UCID: [partner's UCID]

Marking scheme

A	8 marks
B	5 marks
C	5 marks
TOTAL	18 marks

Exercise A: A 4-stage pipeline

Read This First

This exercise is about a 4-stage pipeline for the Exam16 ISA that appeared in earlier lab assignments. Note carefully that the number of stages is four, not five. This is possible for Exam16 because data memory addresses for loads and stores come straight from the Register File, not from the ALU, so execution in the ALU and Data Memory access can be done within a single pipeline stage. An interesting but imperfect design for the 4-stage pipeline is shown in Figure 1.

What to Do, Part I: Hazard Analysis

As you saw in lectures, the 5-stage MIPS pipeline of textbook Figure 7.47 sometimes processes instructions correctly but sometimes fails if there are data or control hazards in the instruction sequences found in Instruction Memory. The computer of Figure 1 doesn't have any multiplexers for forwarding and doesn't have register controls for stalls and/or pipeline flushes, so it would be reasonable to suspect that the circuit might not handle data and/or control hazards correctly.

Data Hazards

Consider the following sequence of instructions, and assume that the most recent updates to registers `x2`–`x7` occur many cycles before the `sub` instruction is fetched:

```
sub     x4, x2, x3
add     x5, x5, x4
add     x6, x6, x4
add     x7, x7, x4
```

Which `add` destination GPRs are guaranteed to receive correct results, and which ones might receive incorrect results? Provide a diagram to support your answer.

Now consider a different sequence of instructions, and assume that the most recent updates to registers `x2`–`x6` occur many cycles before the `lw` instruction is fetched:

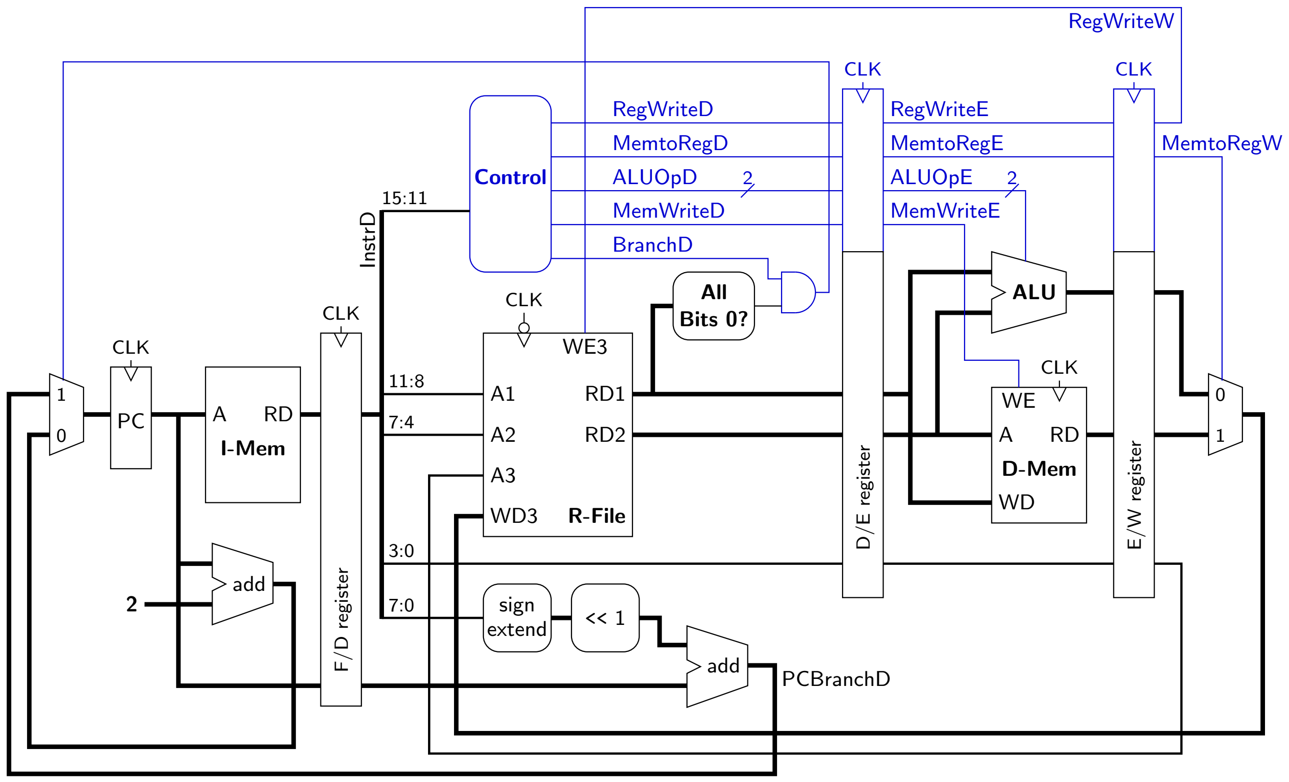
```
lw      x3, (x2)
add     x4, x4, x3
add     x5, x5, x3
add     x6, x6, x3
```

Which `add` destination GPRs are guaranteed to receive correct results, and which ones might receive incorrect results? Provide a diagram to support your answer.

Control Hazards

Consider the following sequence of instructions, and assume that `x9` has a value of 0 when the `brz` instruction is decoded, so the branch should be taken. The circuit will behave incorrectly, because one or more of the last three `add` instructions will get into the pipeline before `lw`, and the circuit has no ability to cancel those instructions.

Figure 1: 4-stage pipeline for the Exam16 ISA. Note that unlike in the 5-stage pipelines in the textbook and lecture notes, execution in the ALU and Data Memory access both occur in the E stage.



```
L1:  lw      x8, (x2)
     add     x2, x2, x3
     add     x4, x4, x8
     sub     x9, x2, x5
     add     x0, x0, x0      # NOP to manage data hazard
     brz     x9, L1
     add     x10, x10, x11
     add     x10, x10, x12
     add     x10, x10, x13
```

Exactly how many incorrect updates to `x10` *will occur? Provide a diagram to support your answer.*

What to Do, Part II: Timing Analysis

Let's determine a maximum clock frequency for the computer of Figure 1, assuming timing parameter values given in Figure 2.

When do timing analysis of a pipelined circuit, each stage can be considered independently of the other stages, and the stage that has the longest minimum clock period will determine the maximum clock frequency for the whole circuit.

E stage

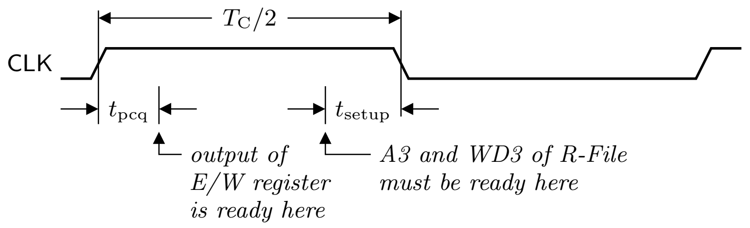
Let's start with the E stage, because it's the easiest to analyze. *What is the minimum clock period for reliable operation of the E stage? Show how you obtained your answer.*

F stage

The F stage is the next easiest to analyze. You need to check both the path from the PC to the F/D register through I-Mem and the path from the PC back to the PC through the adder in the F stage. *What is the minimum clock period for reliable operation of the F stage? Show how you obtained your answer.*

W stage

The W stage is easy to analyze too, but you must take into account that R-File updates occur on negative clock edges. You may assume that negative clock edges occur exactly $T_C/2$ after positive clock edges, where T_C is the clock period. The following sketch may help:


$T_C/2$
CLK
t_{pcq}
t_{setup}
output of E/W register is ready here
A3 and WD3 of R-File must be ready here

What is the minimum clock period for reliable operation of the W stage? Show how you obtained your answer.

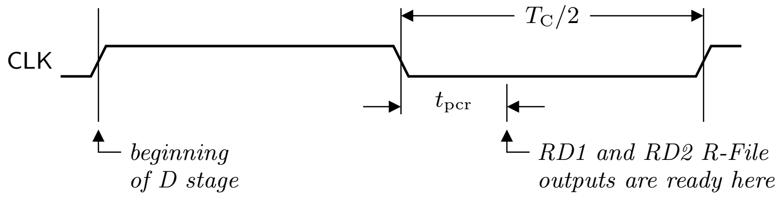
D stage

The timing analysis for the D stage is trickier than it is for the other three stages, because one of the GPRs being read during a clock cycle may have its value written to halfway through that clock cycle. For the purposes of this exercise, let's define

Figure 2: Timing parameters for elements in the computer of Exercise A. All values are in picoseconds. The values are mostly taken from Lab 8, Exercise B, but there is an important difference for R-File reads: The parameter t_{pcr} (clock-to-read-port propagation delay) is the worst-case delay after a *negative* clock edge until both R-File outputs RD1 and RD2 are ready.

element	parameter	value
PC	t_{pcq}	32
	t_{setup}	21
pipeline register	t_{pcq}	32
	t_{setup}	21
I-Mem	t_{pd}	225
Control	t_{pd}	90
R-File	t_{pcr} for read	68
	t_{setup} for write	26
sign extend	t_{pd}	56
<< 1	t_{pd}	40
adder	t_{pd}	170
All Bits 0?	t_{pd}	127
ALU	t_{pd}	194
D-Mem	t_{pd} for read	237
	t_{setup} for write	30
AND gate	t_{pd}	24
2:1 mux	t_{pd}	33

a parameter called t_{pcr} (clock-to-read-port propagation delay), which will be the worst-case delay from a *negative* clock edge to the time when R-File outputs RD1 and RD2 are ready:



There are three potentially critical paths in the D stage:

(1) The path involving RD1, All Bits 0?, the AND gate, and a multiplexer.
(2) The path involving Control, the AND gate, and a multiplexer.
(3) The path involving the branch target address PCBranch.

What is the minimum clock period for reliable operation of the D stage? Show your analysis of the three paths listed above.

Overall maximum clock frequency

Combine your results for the E, F, W, and D stages to determine the maximum clock frequency for reliable operation of the circuit.

What to Include in Your PDF Submission

Answers for Parts I and II.

Exercise B: `beq` in textbook Figure 7.51

Read This First

The processor of textbook Figure 7.51 makes an attempt to handle `beq` instructions, but *does not* handle them correctly. In this exercise, you are asked to determine what exactly the processor does with a `beq`, *not* to guess based on what you think *should* happen.

What to Do

Consider this program fragment, running in the processor of Figure 7.51 with a clock period of 1.0 ns (as in Lab 8, Exercise B):

instruction address	*instruction*	*disassembly*
`0x0040_0078`	`0x8e11_0000`	`L1: lw x17, (x16)`
`0x0040_007c`	`0x0128_0833`	`add x16, x16, x18`
`0x0040_0080`	`0x0000_0013`	`nop`
`0x0040_0084`	`0x0000_0013`	`nop`
`0x0040_0088`	`0x0000_0013`	`nop`
`0x0040_008c`	`0xfe08_86e3`	`beq x17, x0, L1`
`0x0040_0090`	`0x0149_f433`	`and x8, x19, x20`
`0x0040_0094`	`0x0149_e4b3`	`or x9, x19, x20`
`0x0040_0098`	`0x0149_a533`	`slt x10, x19, x20`
`0x0040_00a0`	`0x4149_85b3`	`sub x11, x19, x20`

Suppose that the Fetch stage for the `beq` instruction starts at $t = 90.0$ ns, and suppose that the `lw` instruction has put a value of zero into `x17`.

Answer the following questions. As in Lab 8, Exercise B, use hexadecimal notation for 32-bit numbers, and explain how you got your answers.

1. At $t = 91.0$ ns, what gets written into the PC?
   Shortly after $t = 91.0$ ns, what are the values of InstrD and PC4D?
2. At $t = 92.0$ ns, what gets written into the PC?
   Shortly after $t = 92.0$ ns, what are the values of InstrD and PC4E?
3. Just before $t = 93.0$ ns, what are the values of PCTargetE and ZeroE?
4. At $t = 93.0$ ns, what gets written into the PC?
   Shortly after $t = 93.0$ ns, what is the value of InstrD?
5. At $t = 94.0$ ns, what gets written into the PC?
   Shortly after $t = 94.0$ ns, what is the value of InstrD?

What to Include in Your PDF Submission

Answers to Questions 1–5.

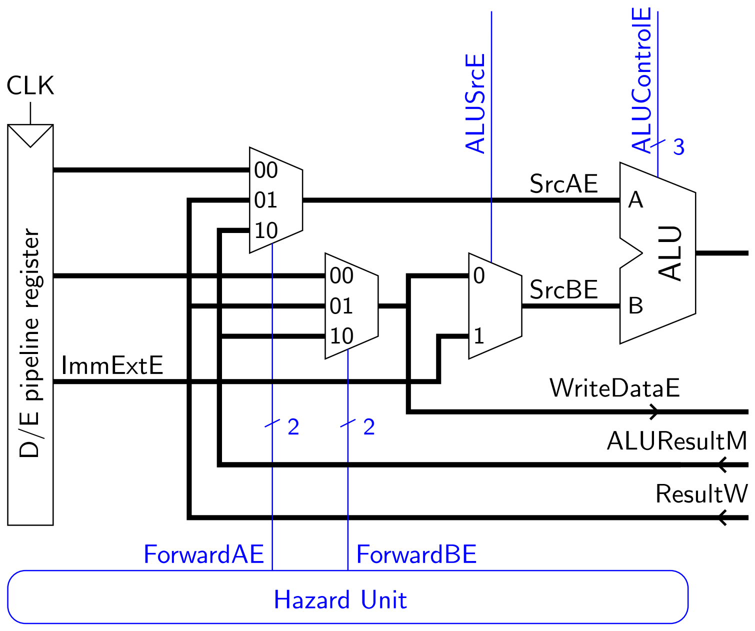
Exercise C: Forwarding details

Read This First

The point of this exercise is to help you understand the forwarding logic presented in Figure 7.54. in your textbook.

Figure 3 shows the outputs of the Hazard Unit, and the multiplexers that guide the correct source data into the ALU.

Figure 3: For Exercise C, diagram to highlight the multiplexers added for forwarding in the Execute stage in textbook Figure 7.55. See the textbook figure for full details, including the *inputs* to the Hazard Unit.



What to Do, Part I

There are four data hazards in the following instruction sequence:

```
sw      x0, 0(x29)     # line  1
sw      x0, 4(x29)     # line  2
sw      x0, 8(x29)     # line  3
add     x25, x16, x17  # line  4
sub     x24, x18, x25  # line  5
lw      x9, (x8)       # line  6
addi    x24, x24, 1    # line  7
add     x10, x10, x9   # line  8
sw      x10, (x11)     # line  9
```

The first hazard is the use of the first `add` result as a source in the `sub` instruction of line 5. Here is a detailed description of how this hazard is managed by the forwarding unit:

> During the Execute stage of `sub`, the Hazard Unit detects that Rs2E (11001_{two} for `x25`) matches RdM (also 11001_{two} for `x25`) and that RegWriteM=1. So it sets ForwardBE=10 so that ALUResultM (the first `add` result) is passed to the "B" input of the ALU.

Identify the other three data hazards. For each hazard, give details of how the Hazard Unit solves the hazard, using the above example as a model.

What to Do, Part II

There is an obvious data hazard in the following instruction sequence:

```
    lw      x10, (x8)
    sw      x10, (x9)
```

This kind of code is very common when data is being copied from one memory location to another memory location.

Consider the circuit of Figure 7.55 in the textbook. For the above instruction sequence, which one of the following things does the circuit do?

- Handle the hazard properly using only forwarding;
- handle the hazard properly using a combination of stalling and forwarding;
- fail to handle the hazard properly, storing an out-of-date `x10` value;
- or fail in a different way, storing some other wrong value?

Give a brief explanation for your answer—just a few short sentences.

What to Include in Your PDF Submission

Answers for Parts I and II.