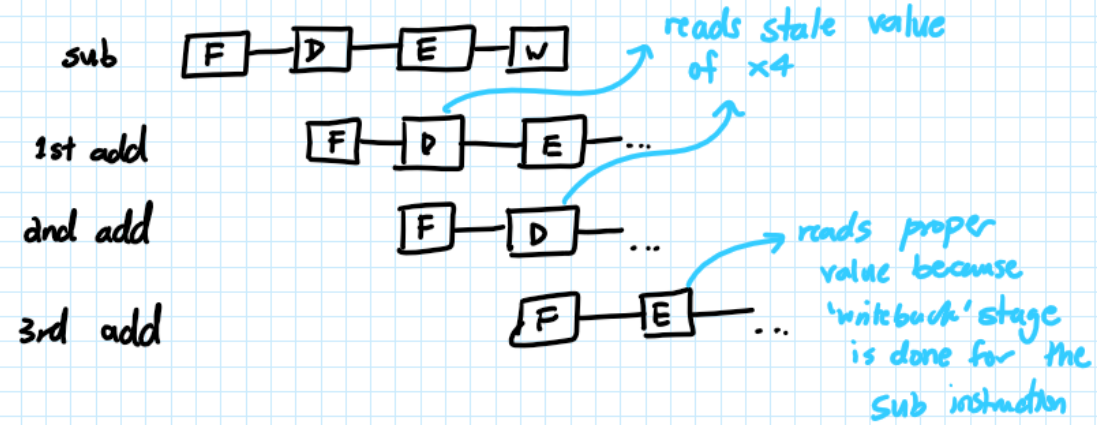
| | |
|---|---|
| **Course:** | Computer Organization - ENCM 369 |
| **Lab #:** | 9 |
| **Instructor:** | N. Bartley |
| **Group Submission for:** | B02 |
| **Submitted by:** | Aarsh Shah |
| **UCID:** | 30158991 |
| **Partner:** | William Fraser |
| **UCID:** | 30150079 |
| **Date Submitted:** | 29-Mar-2023 |

# Exercise A

**Part I:**

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

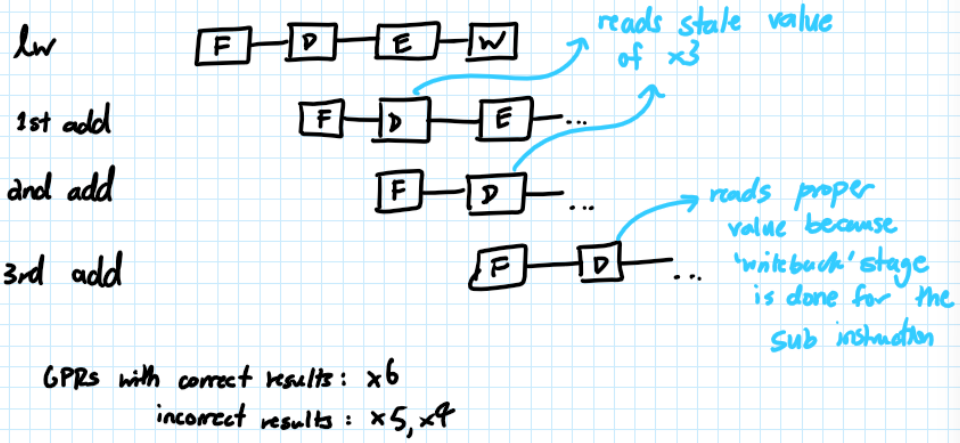
*Which* `add` *destination GPRs are guaranteed to receive correct results, and which ones might receive incorrect results? Provide a diagram to support your answer.*




GPRs with correct results: x7
incorrect results: x5, x6

```
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.*



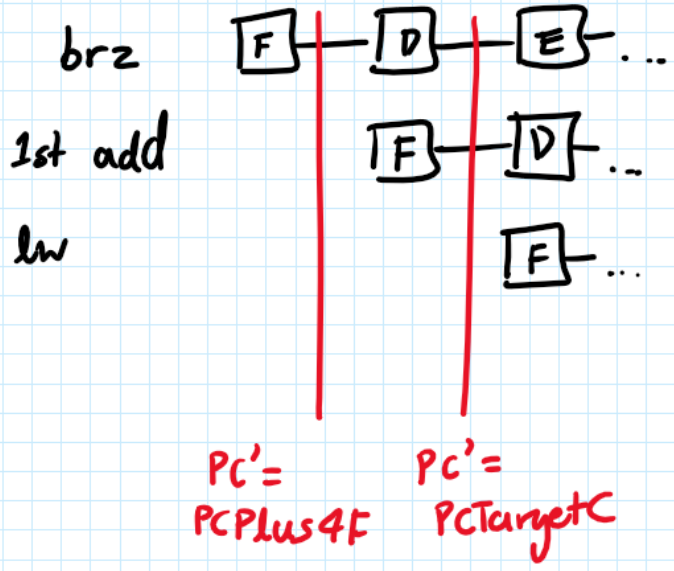

GPRs with correct results: x6
incorrect results: x5, x4

**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.

```
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.*




Thus, there will be 1 incorrect update to GPR x10 because the first add after brz is added to the pipeline immediately after, but the second add won't be because the PC is updated to the correct value after the second stage.

Part 2:

**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.*

Paths through E to check:

1) ALU = 194

2) D-mem = 237

For E stage

$T_c \geq 32 + 237 + 21 = 290$

32: tpcq for pipeline register; 237: worst tpd; 21: t-setup for pipeline register

**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.*

Path's through F-stage

$T_c \geq 32 + 170 + 33 + 21 = 256$

32: tpcq for PC register; 170 + 33: tpd (adder + mux); 21: tsetup for PC register

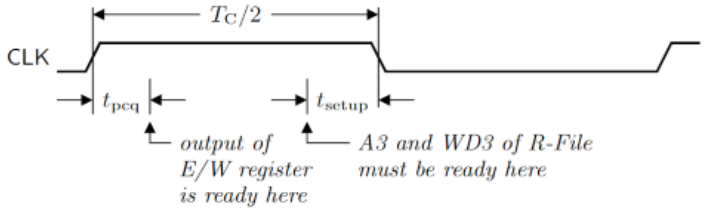
$T_c \geq 32 + 225 + 21 = 278$

32: tpcq from PC; 225: tpd for I-mem; 21: tsetup for pipeline register

Thus $T_c \geq 278$

**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:




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

$T_c \geq$ 32 + 33 + 26 = 91

32: tpcq from E/W pipeline register

33: tpd from 2:1 mux

26: tsetup for R-file

However since this for only half the clock period we must multiply by 2. 91(2) = 182 thus $T_c \geq 182$

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.*

1) $T_c \geq \underbrace{32}_{t_{pcq}\text{ for pipeline register}} + \underbrace{127 + 24 + 33}_{t_{pd}\text{ for All bits 0, AND, and 2:1 mux}} + \underbrace{21}_{t\text{-setup for PC}} = 237$

Here we look at the larger side of the CLK cycle and then multiply by two to get the full cycle. $(237)2 = 474$.

2) $T_c \geq \underbrace{32}_{t_{pcq}\text{ for pipeline register}} + \underbrace{90 + 24 + 23}_{t_{pd}\text{ for control, AND gate, 2:1 mux}} + \underbrace{21}_{t_{setup}\text{ for PC.}} = 190$

3) $T_c \geq \underbrace{32}_{t_{pcq}\text{ for Pipeline register}} + \underbrace{56 + 40 + 170 + 33}_{t_{pd}\text{ for sign ext, <<1, adder and 2:1 mux}} + \underbrace{21}_{t_{setup}\text{ for PC}} = 352$

Thus $T_c \geq 474$

**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.*

Here. $T_c \geq 474$ for reliable operation of the circuit (the largest value above), thus $f \leq \frac{1}{T_c} = \frac{1}{474} = 4.219E-3$

## Exercise B

Exercise B

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?

PC = 0x0040_0090 which is the instruction address of the AND instruction, since the beq doesn't yet know to branch yet.

InstrD = 0xfe08_8be3. This is the instruction that is pulled from the instruction memory for the beq instruction.

PCD = 0x0040_008c. This the instruction address of beq.

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?

PC = 0x 0040_0094. Similar to above, this is the OR instruction, since the beq doesn't know to branch yet.

InstrD = 0x 0199_f433, the instruction that is pulled from the instruction memory for the AND instruction.

PC4E = 0x0090-008c. The beq instruction address since its at the execute stage now.

3. Just before $t = 93.0$ ns, what are the values of PCTargetE and ZeroE?

PCTargetE = 0x0040_0078. The processor has computed where the L1 label is located using the instruction address of beq and the extended value.

zeroE = 1, since the ALU does a comparison between x17 and x0, and since they both hold the value 0, the difference is 0 and thus zeroE is triggered to be 1.

4. At $t = 93.0$ ns, what gets written into the PC?
Shortly after $t = 93.0$ ns, what is the value of InstrD?

PC = 0x0040_0078, the beq now switched the 2:1 mux on the PC to take the PCtargetE value.

InstrD = 0x0149_c963, this is the instruction for the OR instruction getting to the decode stage

5. At $t = 94.0$ ns, what gets written into the PC?
Shortly after $t = 94.0$ ns, what is the value of InstrD?

PC = 0x0040_007c. This is the next instruction after the lw at the L1 label.

InstrD = 0x8e11_0000. This is the instruction for the load word.

# Exercise C

## Part 1:

The second data hazard is the use of the sub result as a source in the addi instruction at line 9. Here is a detailed description of how it is managed by the forwarding unit:

> During the Execute stage of addi, the Hazard Unit detects that Rs1E ($11000_2$ for x24) matches the RdW and that RegWriteW = 1. So, it sets ForwardAE=01 so that ResultW is passed to the A input of the ALU.

The third data hazard is the use of the lw result from line 6 used as a source for the add instruction in line 8. Here is a detailed description of how it is managed by the forwarding unit:

> During the Execute stage of add, the Hazard Unit detects that Rs2E ($01001_2$ for x9) matches RdW and that RegWriteW = 1. So, it sets ForwardBE = 01 so that ResultW is passed to the B input of the ALU.

The final data hazard is the use of the result of the add in line 8 being used as the srouce for the store word in line 9. Here is a detailed description of how it is managed by the forwarding unit:

> During the Execute stage of sw, the Hazard Unit detects that Rs2E ($01010_2$ for x10) matches RdM and that RegWriteM = 1. So, it sets ForwardBE = 10 so that ALUResultM (the add result) will be used as WriteDataE, and in the next clock cycle be WriteDataM, which is the value written to the register specified in the store word instruction (x11).

## Part 2:

The circuit in Figure 7.55 of the textbook will fail to handle the hazard properly and store an out-of-date x10 value. Although in the Execute stage of the sw instruction the Hazard unit would correctly identify that Rs2E ($01010_2$ for x10) matches RdM, and that RegWriteM = 1, therefore detecting a hazard, it would not be able to handle this error. This is because none of the inputs for ForwardBE will be able to give WriteDataE the correct value, so an out-of-date value will be used. It would need the ReadDataW (the result of the lw instruction) from the next clock cycle in order to have the correct result, however, this would require stalling. With this circuit design, this circuit does not account for stalling so the sw instruction will have to proceed past the Execute stage, with no way to fix the value after that. We know this circuit does not account for stalling as it does not have an enable for the PC, among other missing components, so it is unable to stop an instruction from proceeding. Thus, this circuit will fail to handle the hazard properly and store an out-of-date x10 value.