CS 382 Computer Architectu **Homework 3**

Fall 2023 and Organization

Requirement (READ FIRST!)

You have to **type** the solutions. **Handwritten homework will not be graded and will receive zero credit.** You can annotate on this document directly (you might need to know how to insert a picture into a PDF file); or you can submit a separate PDF, but with solutions clearly marked with question numbers.

I pledge my honor that I have abided by the Stevens Honor System.

- Anna Hauk

1 (15 points)

When silicon chips are fabricated, defects in materials (*e.g.,* silicon) and manufacturing errors can result in defective circuits. A very common defect is for one signal wire to get "broken" and always register a logical 0. This is often called a "stuck-at-0" fault. Answer the following questions based on Figure 3.27 in the textbook.

1.1 (5 points)

Which instructions fail to operate correctly if the Br wire is stuck at 0?

B, BL, CBZ

No RET since is's prompted by the PBr wire

1.2 (5 points)

Which instructions fail to operate correctly if the ALUsrc wire is stuck at 0?

ALUsrc chooses between an immediate number and a register read. It will always pick the register value over immediate value so any instructions that use an immediate as the second input would fail.

AND, ORR, SUB, ADD, LDR, STR, ANDS, ADDS, SUBS

1.3 (5 points)

Which instructions fail to operate correctly if the RegWrite wire is stuck at 0?

RegWrite denotes if we write to a register or not. So, any instructions needing to write to registers are affected.

AND, ORR, SUB, ADD, LDR, ANDS, ADDS, SUBS, BL, CBZ, RET

2 (20 points)

Consider adding a new instruction, SWAP Rd, Rn , to our architecture, which will swap the data stored in Rd

and Rn . Discuss the necessary changes that must be made to the datapath above. There is no need to design a new datapath - just discuss what should be added to the existing one.

Since you're writing to 2 registers at once, you'll need a copy of each register. So you can make "temporary registers" to hold it and that means you'd need another WriteReg and RegDataW. You'd also need to be able to choose the correct registers so, you'll need more multiplexors and a control signal to indicate when you need the second WriteReg. You'll have to also make new opcode!

Changes:

- copied registers → more multiplexors
- Need to write to both Rd and Rn → additional WriteReg and RegDataW
- Control signal addition
- Opcode change/addition

3 (15 points)

Consider the addition of a multiplier to the CPU shown in Figure 3.27 in the textbook. This addition will add 200 ps to the latency of the ALU, but will reduce the number of instructions by 20% (because there will no longer be a need to emulate the multiplication instruction). Answer the following questions based on **Chapter 3.3** of the textbook **(no pipelining)** and the following table for the latencies of the stages.

IF	ID	EX	ME	WB
200 ps	250 ps	150 ps	300 ps	200 ps

3.1 (5 points)

What is the clock cycle time with and without this improvement?

Without the improvement the clock cycle is $200 + 250 + 150 + 300 + 200 = 1100ps$

With the improvement $200 + 250 + (150 + 200) + 300 + 200 = 1300ps$

3.2 (5 points)

What is the speedup achieved by adding this improvement? *Hint:* $= 1 - \frac{new\ time}{old\ time}$.

N = original number of instructions. There are 20% less instructions so its (0.8N)

$1 - \frac{0.8 \times 1300}{1100} = 1 - \frac{1040}{1100} = 1 - 0.945 = 0.0545$

3.3 (5 points)

What is the slowest the new ALU can be and still result in improved performance?

$0.8(x) < 1100$ total

$x = \frac{1100}{0.8} = 1375ps$

$1375 - 200 - 250 - 150 - 300 - 200 = 275ps$ ALU slowest

4 (20 points)

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Questions in this problem assume that individual stages of the datapath have the latencies shown in Problem 3 above. Answer the following questions.

4.1 (5 points)

What is the clock cycle time in a pipelined and non-pipelined processor?
Non-pipelined processor is $200 + 250 + 150 + 300 + 200 = 1100ps$ per clock cycle
Pipelined processor 300ps for each clock cycle (max latency)
Pipelined processor is 300 ps for each clock cycle.

4.2 (5 points)

What is the total latency of an LDR instruction in a pipelined and non-pipelined processor?
In a pipelined processor, the total latency of an instruction is determined by the latency of the longest pipeline stage.

Non-pipelined: $200 + 250 + 150 + 300 + 200 = 1100\,ps$

Pipelined: $300 * 5 = 1500\,ps$

4.3 (10 points)

If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

I would split the memory read stage only because it takes the longest time and so by splitting it in half, the max becomes the ID stage which is 250 so it would take max 250ps for each clock cycle instead of 300ps.

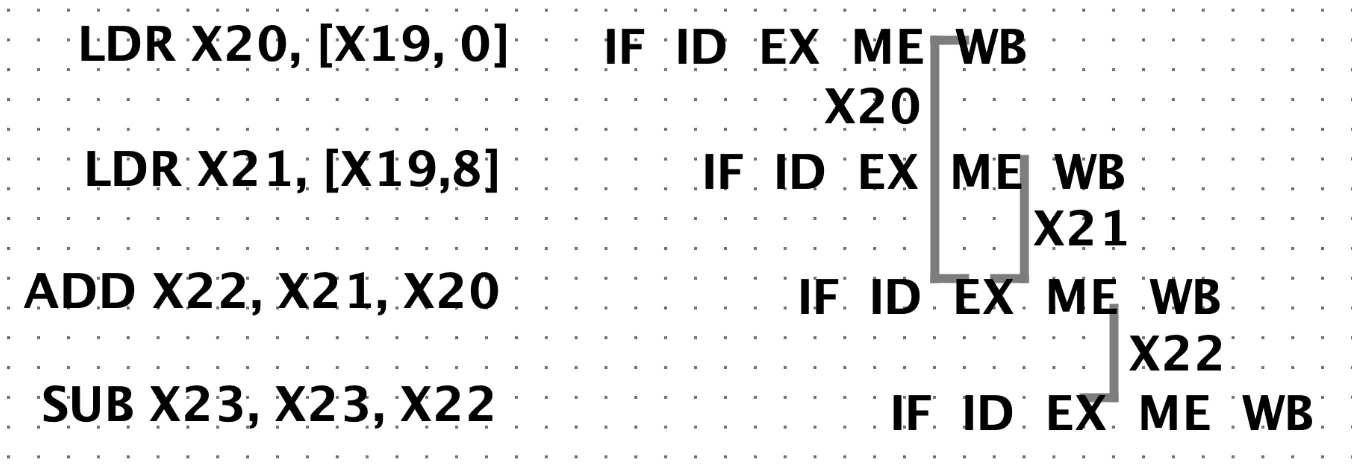
5 (20 points)

Consider the following instructions executed in a pipeline with full forwarding support (including WB write/read in the same cycle). Identify the value of which register is forwarded from a stage of an instruction to a stage of a subsequent instruction. (Completely impossible example: “The value of X5 is forwarded from the IF stage of instruction 1 to the WB stage of instruction 2.”) Submit a pipeline execution diagram as shown in the textbook.

```
LDR X20, [X19, 0]
LDR X21, [X19, 8]
ADD X22, X21, X20
SUB X23, X23, X22
```

If instruction at ME stage is going to write to a register thats the same as the read register at the EX stage, then we have a data hazard. Never forward data to ID stage, always EX stage. Source: ME or WB stage. Destination: Always EX.
So:

1. Forward X20 from WB to EX in ADD instruction
2. Forward X21 from ME to EX in ADD instruction
3. Forward X22 from ME to EX in SUB instruction



6 (20 points)

Consider the following instructions.

```
LDR   X1, [X6, 8]
ADD   X0, X1, X0
STR   X0, [X10, 4]
LDR   X2, [X6, 12]
SUB   X3, X0, X2
STR   X3, [X8, 24]
CBZ   X2, 40
```

6.1 (10 points)

Add NOP instruction to the code above so that it will run correctly on a pipeline **without forwarding**, but WB write/read in the same cycle. (A pipeline execution diagram is not needed for this problem. Sketching it may help, but it will not be graded.) WB and ID need to line up

```
LDR   X1, [X6, 8]
NOP
NOP
ADD   X0, X1, X0
NOP
NOP
STR   X0, [X10, 4]
LDR   X2, [X6, 12]
NOP
NOP
SUB   X3, X0, X2
NOP
NOP
STR   X3, [X8, 24]
CBZ   X2, 40
```

X1 issue

IF	ID	EX	ME	WB		
NOP1	IF	ID	EX	ME	WB	
NOP2		IF	ID	EX	ME	WB
ADD			IF	ID	EX	ME

X0 issue

IF	ID	EX	ME	WB		
NOP1	IF	ID	EX	ME	WB	
NOP2		IF	ID	EX	ME	WB

STR			IF	ID	EX	ME

6.2 (10 points)

Optimize the code execution by re-arranging the instructions to get the same correct result faster

```
LDR X1, [X6, 8]
LDR X2, [X6, 12]
NOP
ADD X0, X1, X0
NOP
NOP
SUB X3, X0, X2
STR X0, [X10,4]
NOP
STR X3, [X8, 24]
CBZ X2, 40
```

Original:

```
LDR    X1, [X6, 8]
ADD    X0, X1, X0
STR    X0, [X10, 4]
LDR    X2, [X6, 12]
SUB    X3, X0, X2
STR    X3, [X8, 24]
CBZ    X2, 40
```