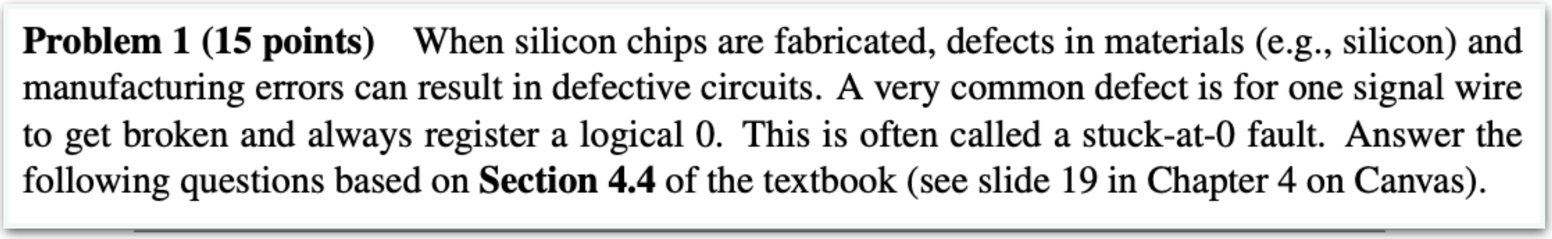
Homework 3



Problem 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 **Section 4.4** of the textbook (see slide 19 in Chapter 4 on Canvas).

Problem 1—

A. Which instructions fail to operate correctly if the MemToReg wire is stuck at 0?

When MemToReg stuck at 0, it is unable to read from memory…

LDUR Instructions will fail because they must read value from memory

B. Which instructions fail to operate correctly if the ALUSrc wire is stuck at 0?

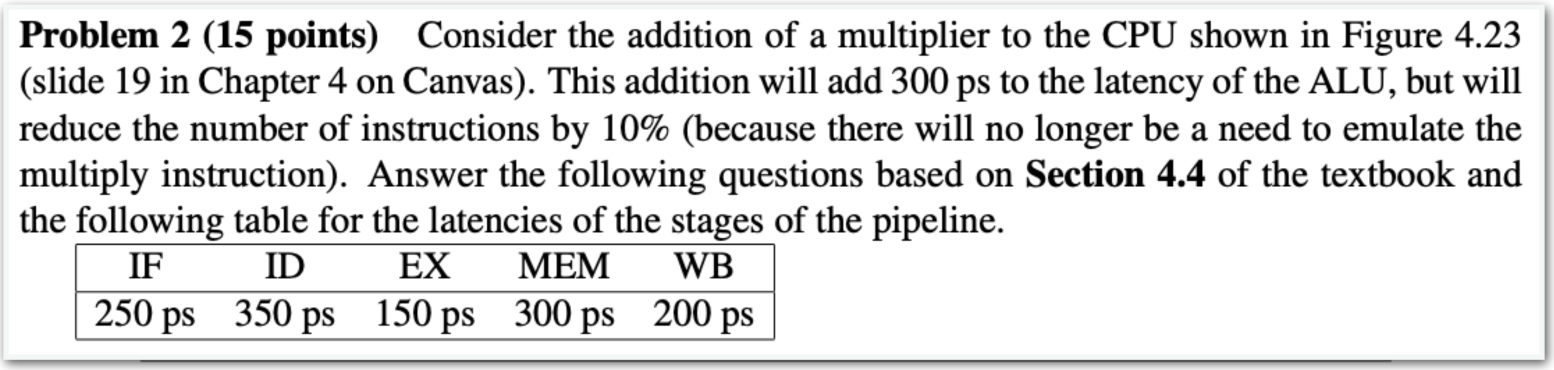
While stuck at 0, you cannot get the value from any instructions, only the available registers…

I-instructions fail including STUR, LDUR, ORR

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

When stuck at 0, no register can be written in…

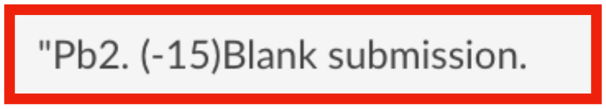
Instructions that fail include arithmetic instructions & LDUR because you cannot write the value in



Problem 2 (15 points) Consider the addition of a multiplier to the CPU shown in Figure 4.23 (slide 19 in Chapter 4 on Canvas). This addition will add 300 ps to the latency of the ALU, but will reduce the number of instructions by 10% (because there will no longer be a need to emulate the multiply instruction). Answer the following questions based on **Section 4.4** of the textbook and the following table for the latencies of the stages of the pipeline.

IF	ID	EX	MEM	WB
250 ps	350 ps	150 ps	300 ps	200 ps

Problem 2—

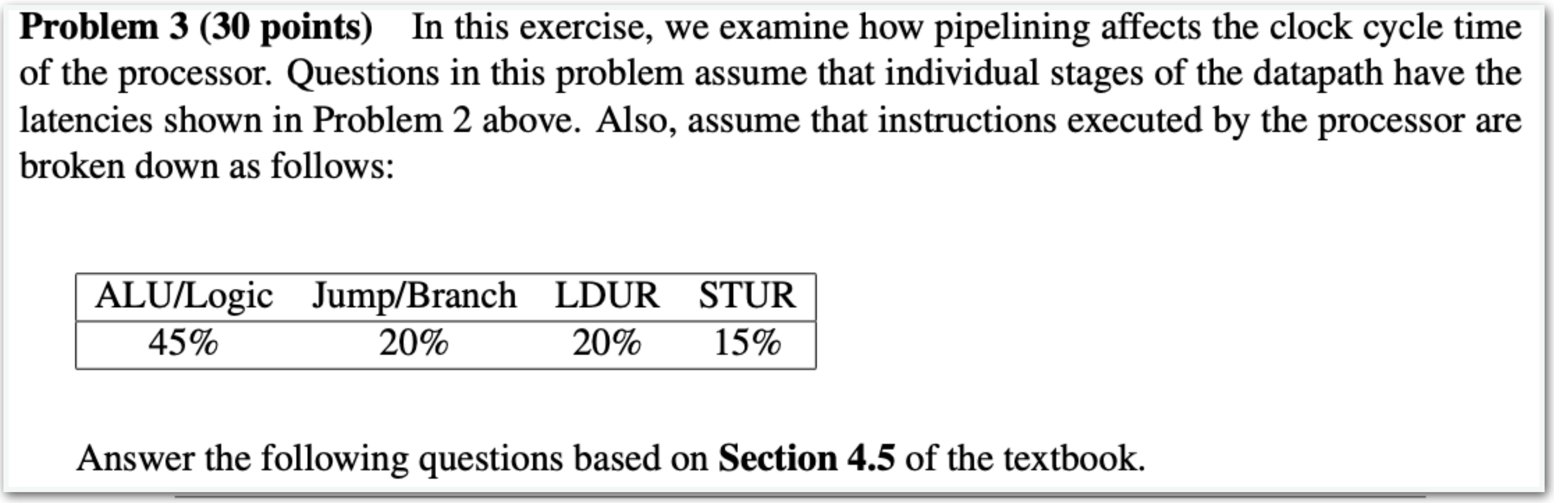


"Pb2. (-15)Blank submission.

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

B. What is the speedup achieved by adding this improvement?

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



Problem 3 (30 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 2 above. Also, assume that instructions executed by the processor are broken down as follows:

ALU/Logic	Jump/Branch	LDUR	STUR
45%	20%	20%	15%

Answer the following questions based on **Section 4.5** of the textbook.

Problem 3—

A. What is the clock cycle time in a pipelined and non-pipelined processor?

Pipeline needs to account for the slowest and can use the ID that **= *350ps***

Non-Pipeline counts all stages: 250+350+150+300+200 **= *1250ps***

B. What is the total latency of an LDUR instruction in a pipelined and non-pipelined processor?

Pipeline: LDUR = 20% which would account for 5 cycles while the ID is 350ps….

5 x 350ps **= *1750ps***

Non-Pipeline: 250+350+150+300+200 **= 1250ps**

C. 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?

largest stage = 350ps

350/2 = 175ps

now longest is 300ps so new clock cycle time **= *300ps***

D. Assuming there are no stalls or hazards, what is the utilization of the data memory?

data memory is only being used currently by LDUR ad STUR which account for 20% and 15%

data memory utilization = 20 + 15 **= *35%***

Problem 4 (10 points) Consider the following loop.

```
LOOP: LDUR X10, [X1, #0]
      LDUR X11, [X1, #8]
      SUB X13, X11, X10
      ADD X12, X12, X13
      SUBI X1, X1, #16
      CBNZ X12, LOOP
```

Assume that perfect branch prediction is used (no stalls due to control hazards) and that the pipeline has full forwarding support. Show a pipeline execution diagram for the first two iterations of this loop based on **Section 4.7** of the textbook.

Making a table is not required. Just align stages of difference instructions that take place concurrently.

Problem 4 — **Assume that perfect branch prediction is used (no stalls due to control hazards) and that the pipeline has full forwarding support. Show a pipeline execution diagram for the first two iterations of this loop based on Section 4.7 of the textbook. Making a table is not required. Just align stages of difference instructions that take place concurrently.**

LDUR and LDUR forward data to SUB

SUB forwards data to ADD

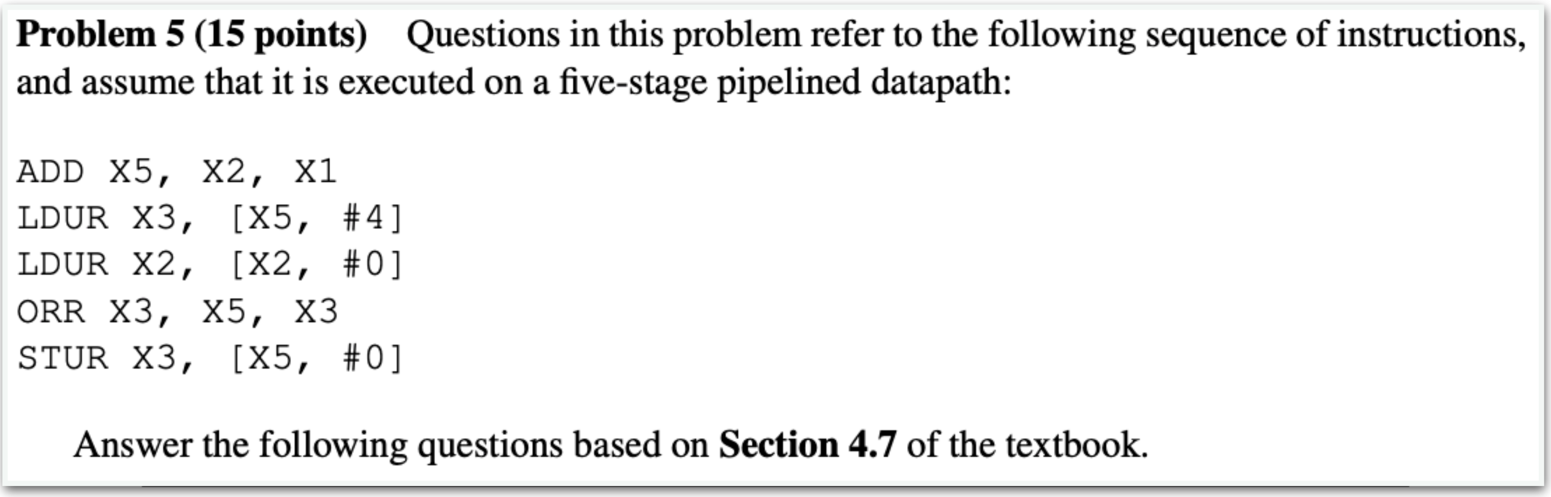
ADD forwards data to CBNZ

LDUR and LDUR forward data to SUB

SUB forwards data to ADD

ADD forwards data to CBNZ

Pb4. (-10)This is not a pipeline execution diagram.



Problem 5 (15 points) Questions in this problem refer to the following sequence of instructions, and assume that it is executed on a five-stage pipelined datapath:

```
ADD X5, X2, X1
LDUR X3, [X5, #4]
LDUR X2, [X2, #0]
ORR X3, X5, X3
STUR X3, [X5, #0]
```

Answer the following questions based on **Section 4.7** of the textbook.

Problem 5—

A. If there is no forwarding or hazard detection, insert NOPs to ensure correct execution.

```
ADD x5, x2, x1          // line 1
NOP
NOP
LDUR x3, [x5, #4]       // line 2
LDUR x2, [x2, #0]       // line 3
NOP
ORR x3, x5, x3          // line 4
NOP
NOP
STUR x3, [x5, #0]       // line 5
```

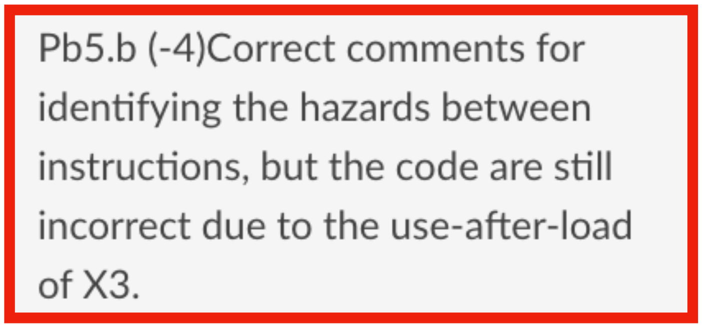
B. Now, change and/or rearrange the code to minimize the number of NOPs needed. You can assume register X7 can be used to hold temporary values in your modified code.

line 2 depends on line 1 for x5 // line 4 depends on line 2 for x3 // line 5 depends on line 4 for x3

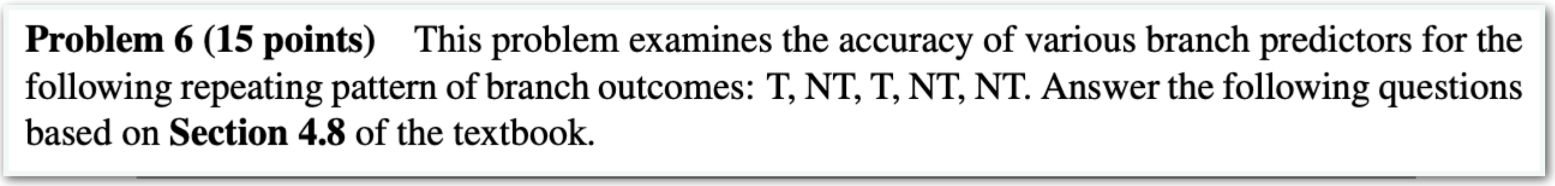
only no depended on line is line 3… so we could potentially move it around without changing code..

moved it up to place of first NOP …. Not sure if this is legal because no x7

```
ADD x5, x2, x1
LDUR x2, [x2, #0]
NOP
LDUR x3, [x5, #4]
NOP
ORR x3, x5, x3
NOP
NOP
STUR x3, [x5, #0]
```



Pb5.b (-4)Correct comments for identifying the hazards between instructions, but the code are still incorrect due to the use-after-load of X3.



Problem 6 (15 points) This problem examines the accuracy of various branch predictors for the following repeating pattern of branch outcomes: T, NT, T, NT, NT. Answer the following questions based on **Section 4.8** of the textbook.

Problem 6—

A. What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

SEQ: T, NT, T, NT, NT

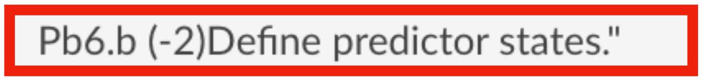
ALWAYS-TAKEN:	T, T, T, T, T => 2/5 = 0.4 = ***40% accurate***
ALWAYS-NOT-TAKEN:	NT, NT, NT, NT, NT => 3/5 = 0.6 = ***60% accurate***

B. What is the accuracy of the 2-bit predictor for this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.62 (predict not taken)? Show the state of the predictor at each step.

SEQ: T, NT, T, NT, NT

using more accurate ALWAYS-NOT prediction

SEQ STATES: 00 - 01 - 10 - 11 - 10 - 10



Pb6.b (-2)Define predictor states."

60%