CS 382: Homework Assignment 3
Due: November 12, 11:55pm

Collaboration Policy. Homeworks will be done **individually**: each student must submit their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. If some code was shown in class, it can be used, but it must be obtained from Canvas, the instructor or the TA.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized.

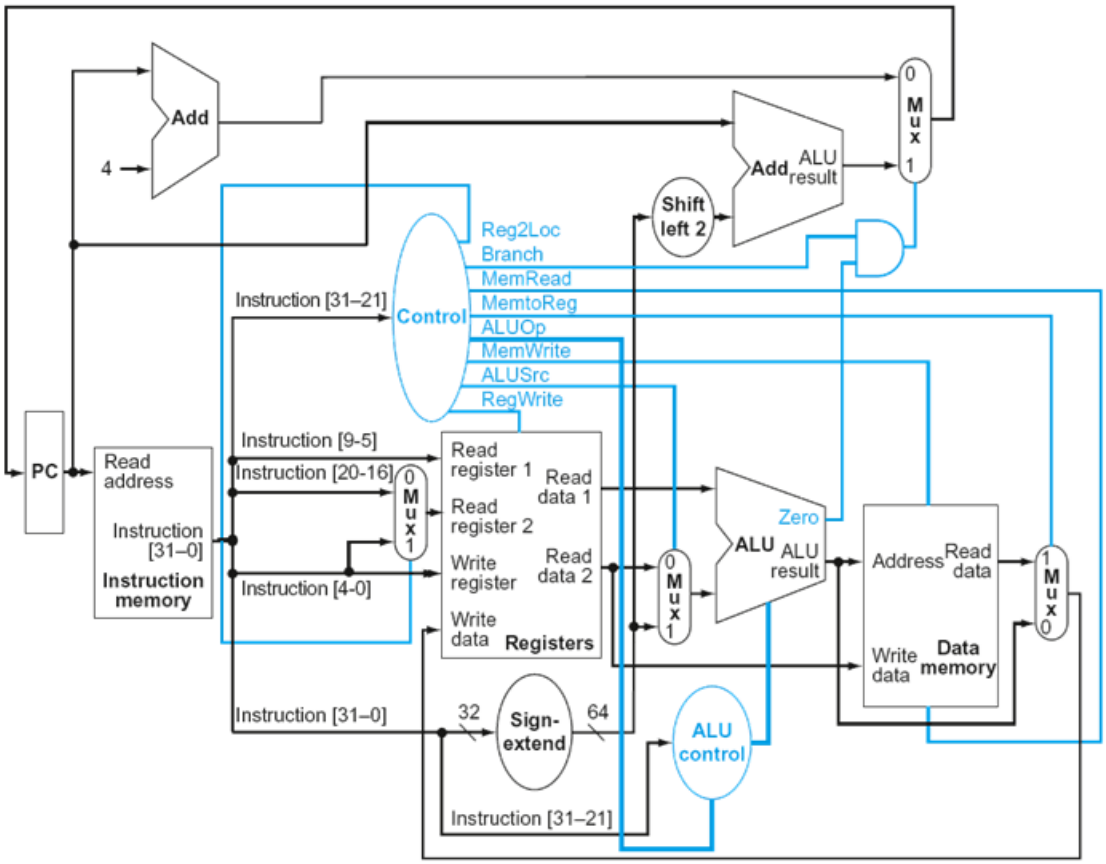
Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

Deliverable. A single **pdf** file on Canvas.

Notes. The weight of each problem is in brackets. **Pay attention to the section of the textbook each problem is based on. The key difference in most cases is whether the processor is pipelined or not.**

Instruction Set For the problems below, unless otherwise specified, assume that the instruction set includes: (i) R-type instructions (ADD, SUB, AND, ORR), (ii) LDUR, (iii) STUR, (iv) CBZ, and (v) B.

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 and the figure below.



(a) [5] Which instructions fail to operate correctly if the Branch wire is stuck at 0?

(b) [5] Which instructions fail to operate correctly if the ALUSrc wire is stuck at 0?

(c) [5] Which instructions fail to operate correctly if the RegWrite wire is stuck at 0?

Problem 2 (10 points) Consider adding a new instruction, `swap Rd, Rn`, to LEGv8. The instruction should implement:

`Reg[Rd]=Reg[Rn]; Reg[Rn]=Reg[Rd]`

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.

Problem 3 (15 points) Consider the addition of a multiplier to the CPU shown above. 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 multiply instruction). Answer the following questions based on **Section 4.4** of the textbook **(no pipelining)** and the following table for the latencies of the stages of the pipeline.

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

(a) [5] What is the clock cycle time with and without this improvement?

(b) [5] What is the speedup achieved by adding this improvement?

(c) [5] What is the slowest the new ALU can be and still result in improved performance?

Problem 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 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.

(a) [5] What is the clock cycle time in a pipelined and non-pipelined processor?

(b) [5] What is the total latency of an LDUR instruction in a pipelined and non-pipelined processor?

(c) [10] 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?

Problem 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 in the **traditional** form (see notes) using the abbreviated names of the stages.

```
1: LDUR X20, [X19, #0]
2: LDUR X21, [X19, #8]
3: ADD X22, X21, X20
4: SUB X23, X23, X22
```

Problem 6 (20 points) Consider the following instructions.

```
LDUR X1, [X6, #8]
ADD X0, X1, X0
STUR X0, [X10, #4]
LDUR X2, [X6, #12]
```

```
SUB X3, X0, X2
STUR X3, [X8, #24]
CBZ X2, 40
```

(a) [10] Add NOP instruction (or bubbles) 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.)

(b) [10] Optimize the code execution by re-arranging the instructions to get the same correct result faster.