Istanbul Technical University

Department of Computer Engineering

BLG 322E - COMPUTER ARCHITECTURE

Assignment 2

Due Date: Thursday, April 10, 2025, 23:59.

- Please draw the diagrams using a **computer program**. PLEASE BE NEAT! If we cannot read or follow your solution, no partial credit will be given.
- Please **type your full name** (first name and last name) **and Student ID** at the top of your solution.
- Please show ALL your work. Answers with no supporting explanations or work will not receive any partial credit. Your homework is not just a final report of your results; we want to see your steps. Upload all the papers you worked on to get to the solution.
- **Submissions:** Submit your solution as a **PDF file** to Ninova before the deadline.
- **No late submissions** will be accepted. Do not send your solutions by e-mail. We will only accept files that have been uploaded to the official Ninova e-learning system before the deadline. Do not risk leaving your submission to the last few minutes.
- **Consequences of plagiarism/cheating:** Assignments have to be done individually. Any cheating will be subject to disciplinary action.

If you have any questions, please e-mail **Berkay Sancı (sancib17@itu.edu.tr)**.

QUESTION 1 (70 POINTS):

Consider the exemplary RISC processor given in Section 2.4.2 of the lecture notes that has an instruction pipeline with the following 5 stages:

- **IF**: Instruction fetch
- **DR**: Instruction Decode, Read registers
- **EX**: Execute
- **ME**: Memory
- **WB**: Write back

Assume the following:

- The CPU has **a single forwarding (bypass) connection** only between the output of the **ME stage (ME/WB register)** and the inputs of the ALU. Both the result from ALU and the result from memory can be forwarded.
- The register file access hazard **is fixed**, i.e., the CPU writes data to registers in the **first** half of the cycle (rising edge) and reads data from registers in the **second** half of the cycle (falling edge).
- The internal structure of the execution circuitry is as shown on slide 2.53 (latest version, 2021), i.e., **branch target address calculation and decision operations** are performed in the **EX** stage, and results are sent directly to the **IF** stage.

```
        XOR     R2, R2, R1
        ADD     R1, $09, R3
        ADD     R1, $03, R4
LOOP:   ADD     R5, R4, R5
        SUB     R3, $03, R6
        STL     $08(R6), R5
        SUB     R4, $01, R4
        BNZ     LOOP
        ADD     R6, $01, R6
        BRU     FINISH
        XOR     R5, $05, R5
        STL     $03(R7), R1
FINISH  LDL     $08(R4), R1
```

a) **[25 pts]** We execute the program given on the left in this instruction pipeline. Draw the space-time diagram for the execution of this program. Solve all data and branch conflicts using software-based NOOP instructions. For the given piece of code, what is the total amount of penalty in clock cycles caused by conflicts?

b) **[10 pts]** What would the total amount of penalty be if the number of iterations was equal to n? (Hint: Your answer should be a function of n). Explain your work/calculations.

c) **[10 pts]** What would be the CPI for the given program?

d) **[25 pts]** Minimize the penalty using optimized software-based solutions, if possible. Keep in mind that the results generated by the program cannot be changed. Draw the new space-time diagram that results from the introduction of these solutions, and explain your solutions. What is the total amount of penalty in clock cycles with these new solutions?

Do not forget to draw the diagrams using a **computer program**!!!

INSTRUCTION SET:

LDL	X(Rs), Rd	Rd ← M[Rs + X]	Load
STL	X(Rs), Rm	M[Rs + X] ← Rm	Store
ADD	Ri, Rj, Rd	Rd ← Ri + Rj	Add
SUB	Ri, Rj, Rd	Rd ← Ri - Rj	Subtract
XOR	Ri, Rj, Rd	Rd ← Ri ⊕ Rj	Exclusive OR
BNZ	Y	PC ← PC + Y	Branch if not zero (Relative)
BRU	Y	PC ← PC + Y	Branch relative

QUESTION 2 (30 POINTS):

A CPU has an instruction pipeline that includes branch prediction mechanisms.

This CPU runs the given piece of code below. In the loop, there is a **branch instruction (BRZ)** that jumps to label L1 if the value of the Counter is divisible by 4.

```
                Counter ← 40; // Initialize Counter by 40
LOOP:           ------
                Counter MOD 4; // Modulo operation
                BRZ L1; // Branch L1 if zero
                ------
L1:             ------
                Counter ← Counter – 1; // Decrement Counter by 1
                BNZ LOOP; // Branch LOOP if not zero
                ------
```

Give the number of correct predictions and mispredictions only for the branch instruction (BRZ) in the loop, if the following branch prediction mechanisms are used. **Explain your results**. Assume that the branch history table includes the branch target address at the beginning.

a) [10 pts] Dynamic prediction with one bit, the initial decision is not to take the branch.

b) [10 pts] Dynamic prediction with two bits, the initial decision is to take the branch (10).

c) [10 pts] Dynamic prediction with two bits, the initial decision is not to take the branch (01).