İTÜ Computer Engineering Department
April 22, 2021
Assoc.Prof. Feza BUZLUCA
Asst. Prof. Sanem KABADAYI

COMPUTER ARCHITECTURE MIDTERM EXAM QUESTION 2

Rules:

1. The normal end time of this question is **18:35**. However, because of the 10-minute buffer period, the submission will be closed at **18:45**.
2. **Answer the question on its own sheet** and upload your files during the time allotted for that question, as explained in the file "Exam policies". Create your files in **PDF format**.
3. **You may not ask any questions during the exam.** State any assumptions you have to make.
4. Any cheating or any attempt to cheat will be subject to the University disciplinary proceedings.
5. Please **show ALL work**. Answers with no supporting explanations or work will be given no partial credit. If we cannot read or follow your solution, no partial credit will be given. PLEASE BE NEAT!

QUESTION 2: (35 Points)

Consider the exemplary RISC processor given in Section 2.4.2 of the lecture notes. **Differing** from lecture notes, suppose that the instruction pipeline is designed with four stages as explained below:

1. **Instruction Fetch and Decode (ID):** Instruction fetch and instruction decode.
2. **Read registers (RR):** Read registers (operands).
3. **Execute (EX):** Perform ALU operation, compute jump/branch targets, make branch decisions.
4. **Memory, Write back (MW):** Two stages in the original processor have been combined; the ME/WB register has been removed.

Assume the following:

- The register file access hazard is **NOT** fixed.
- The processor does **NOT** have any **forwarding (bypass)** connections.
- The internal structure of the execution stage is similar to the circuitry 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 **ID** stage.

a) Draw the timing diagram for the piece of code given below to show the data and branch hazards, and software-based solutions that involve inserting NOOP instructions. (20 p)

```
                ……
1:              SUB  R1, R2, R1        ; R1 <- R1 + R2
2:              LDL  0(R1), R4         ; R4 <- M[R1]
3:              LDL  4(R1), R5         ; R5 <- M[R1+4]
4:              ADD  R6, R4, R6        ; R6 <- R6+R4
5:              ADD  R7, R5, R7        ; R7 <- R7+R5
6:              SUB  R6, R7, R7        ; R7 <- R7 – R6
7:              BNE  LABEL             ; Branch if not equal
8:              ADD  R6, #2, R8        ; R8 <- R6+2
9:              STL  8(R1), R8         ; M[R1+8]<-R8
                  :                    ; other instructions
                  :
10:  LABEL:  …..                       ; program continues
```

b) Assume that there is operand forwarding (bypassing) from the output of the EX stage to the inputs of the ALU. Other parts of the CPU are not modified. Consider only the lines between 1 and 6 (including 6) of the piece of code given in **Part (a)**. Draw the timing diagram for the piece of code (lines 1-6) to show hazards, and software-based solutions that involve inserting NOOP instructions. (15 p)