

EXAM 2

Instructions

1. This is an open books/notes exam.
2. It comprises 20 multiple-choice questions, each worth 5 points.
3. Each question has only one correct answer.
4. For each question, indicate your answer on the answer sheet provided.
5. You have 80 minutes to complete the exam.
6. Please make sure you enter your name in the space below AND on the answer sheet.
7. By signing below, you certify that your conduct throughout the exam has been in accordance with the College of Engineering Honor Code.
8. At the end of the exam, please turn in this exam booklet AND your answer sheet.

Name _____

Signature _____

Performance [questions 1—6]

1. Consider a 5-stage FDExMemWB pipeline in which Fetch takes 2 cycles (i.e., given an address, the instruction memory takes 2 cycles to output the instruction). All other stages can execute in 1 cycle. Assuming perfect branch prediction, full data forwarding, and no load hazards, what is the minimum achievable CPI?
 - a. 3
 - b. 2
 - c. 1.2
 - d. 1
 - e. 0.8

2. When comparing a 5-stage pipeline with its corresponding 5-stage multicycle datapath, which of the following statements are correct?
 - I. The throughput of the pipelined datapath is at least as high
 - II. Individual instructions complete faster on pipelined datapath
 - III. The pipelined datapath requires more hardware resources
 - a. II only
 - b. I and II
 - c. II and III
 - d. I and III
 - e. I, II, and III

3. Consider two versions of a 5-stage pipeline FDExMemWB with full data forwarding that stalls on branches. Version A resolves branches in the MEM stage and has clock period T . Version B resolves branches in EX and has clock period $1.25 \times T$. Which version yields the shortest runtime for a program with 20% branch instructions and no load hazards?
 - a. Version A
 - b. Version B
 - c. Both versions achieve equal times.
 - d. Depends on clock period T .
 - e. Depends on program length.

4. Which of the following can be used to improve pipeline performance in the presence of data hazards?

- I. code reordering
- II. delayed branches
- III. branch prediction
- IV. data forwarding
- V. stalling
- VI. squashing

- a. I and II
- b. I and IV
- c. II and III
- d. I, IV, and V
- e. I, IV, and VI

5. Consider a block of code with 50 assembly instructions (no branches). A compiler augments this code with 10 noop's so that it runs without any stalling on BC2K2, a 15-stage pipelined datapath. How many cycles does it take for the augmented code to run on BC2K2?

- a. 64
- b. 65
- c. 74
- d. 75
- e. None of the above

6. Consider a loop with a total of 60 assembly instructions. A compiler augments it with 15 noop's so that it runs without any stalling on BC2K2, a 15-stage pipelined datapath with "always take" branch prediction. If the loop iterates 2^{20} times, what is the achieved CPI (approximately)?

- a. 0.20
- b. 0.25
- c. 1.25
- d. 4
- e. 5

Hazards [questions 7—8]

Questions 7 and 8 refer to the following piece of LC2K2 assembly and the 5-stage pipeline covered in class. This pipeline has full forwarding to the EX stage. The register file correctly resolves writes/reads to the same register on the same cycle. Branches are resolved in the MEM stage.

```
start  add  4   3   4
        sw   0   4   loc
        lw   4   5   num
        nand 4   5   3
        beq  0   3   start
        add  4   5   4
        nand 3   4   4
        halt
loc    .fill 0
num    .fill 12
```

7. Assuming “branch not taken” is used for prediction, how many REAL data hazards are there in the program? REAL means that we must do either forwarding or stall to resolve the hazard.

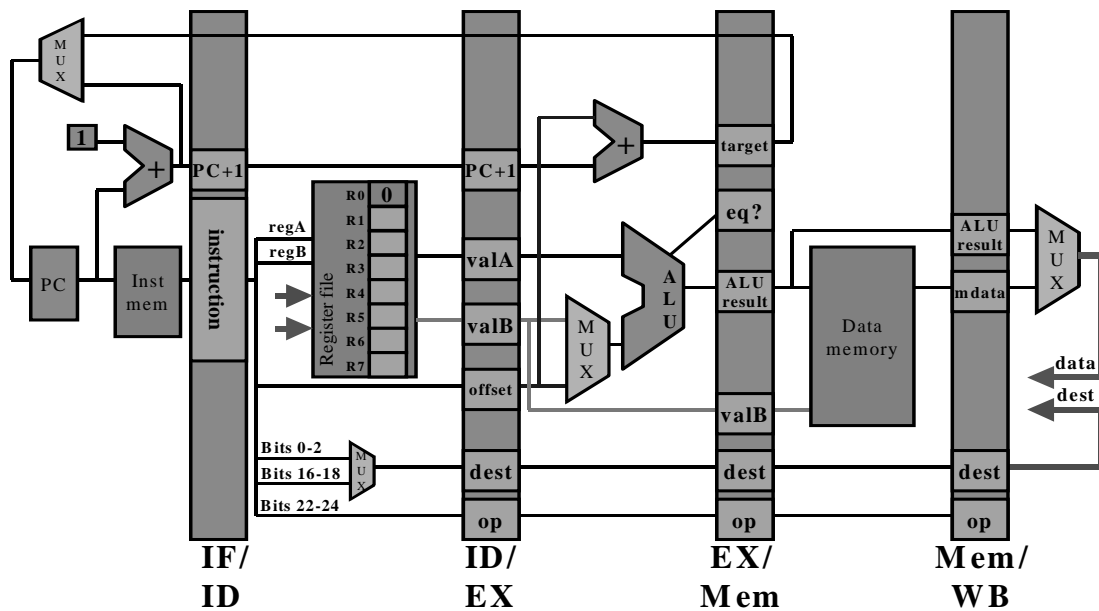
- a. 3
- b. 4
- c. 5
- d. 6
- e. 7

8. Assuming “branch taken” is used for prediction, how many REAL hazards exist in the program?

- a. 3
- b. 4
- c. 5
- d. 6
- e. 7

JALR [questions 9—11]

Consider the familiar LC2K2 instruction `jalr regA regB` which saves PC+1 into regB and branches to the address in regA. (If regA is the same as regB, the processor ends up branching to PC+1.) You wish to implement `jalr` on the following 5-stage pipeline from lecture. (The register file correctly resolves writes/reads to the same register in same cycle.)



9. Your implementation should update both regB and PC during the WB stage. Assuming that the only state elements you can modify are the pipeline registers, which of the following changes to the original pipeline registers are necessary?

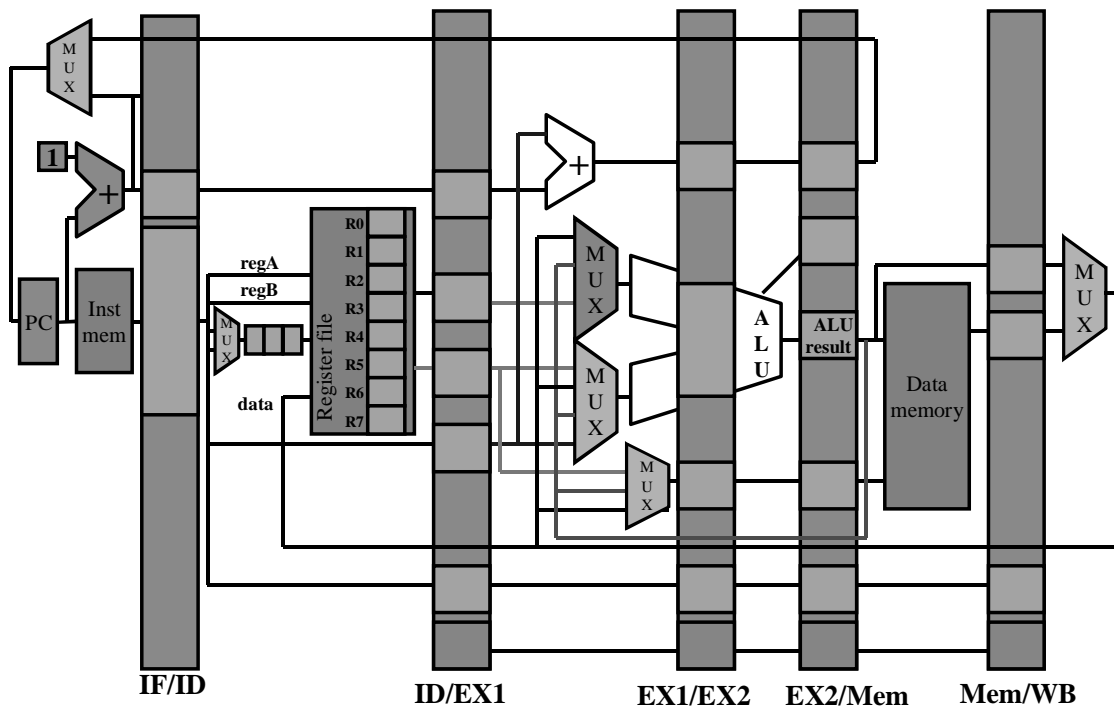
- I. Ex/Mem: Add PC+1 field
 - II. Mem/WB: Add PC+1 field
 - III. Mem/WB: Add eq? field
 - IV. Mem/WB: Add valB field
- a. I and II
 - b. I, II, and III
 - c. I, II, and IV
 - d. III and IV
 - e. II, III, and IV

10. You have discovered an ingenious scheme to update regB during the EX stage (instead of WB) without affecting the clock period of the datapath. PC is still updated during WB. Which of the following statements is correct about the implications of your scheme on pipeline throughput?
- It will improve pipeline throughput, because it speeds up the execution of jalr.
 - It will sometimes degrade pipeline throughput, because it may result in resource contention and cause the pipeline to stall.
 - It will never improve pipeline throughput, because jalr will still take 5 cycles to complete.
 - It will have no effect on pipeline throughput, regardless of the program executed.
 - Cannot tell from the information given.
11. You have discovered another ingenious scheme to update PC during the EX stage (instead of WB) without affecting the clock period of the datapath. Register regB is still updated during WB. Which of the following statements is correct about the implications of your scheme on pipeline throughput?
- It will improve pipeline throughput, because it speeds up the execution of jalr.
 - It will improve pipeline throughput, because it results in fewer instructions being squashed or in fewer noop's.
 - It will never improve pipeline throughput, because jalr will still take 5 cycles to complete.
 - It will sometimes degrade pipeline throughput, because it may result in resource contention.
 - Cannot tell from the information given.

Multistage Pipelines [questions 12—20]

To improve the performance of the 5-stage pipeline from lecture, you consider breaking the EX stage into two stages EX1 and EX2 by introducing an additional pipeline register EX1|EX2. The resulting 6-stage pipeline, shown in the figure below, has the following characteristics.

- Full forwarding from EX2|MEM and MEM|WB registers to EX1 stage
- Register file performs forwarding when the same register is written and read during a clock cycle
- Branches predicted “never taken”
- Branches resolved in MEM stage
- No support for stalling (compiler must insert noop’s to given code whenever required such as, for example, in the case of lw hazards)



12. In comparison with the original 5-stage pipeline, how is the CPI of a program affected by the introduction of the EX1|EX2 pipeline register?
- CPI will not decrease.
 - CPI will not increase.
 - CPI will definitely increase.
 - CPI will remain unchanged.
 - CPI will increase or decrease, depending on the program and the clock periods of the two pipelines.
13. In comparison with the original 5-stage pipeline, how is the branch misprediction penalty affected by the introduction of the EX1|EX2 pipeline register?
- More cycles wasted.
 - Fewer cycles wasted.
 - Same number of cycles wasted.
 - Number of wasted cycles depends on the program.
 - Cannot tell from the given information.
14. In comparison with the original 5-stage pipeline, what do you expect to be the impact of your architectural modification on program runtime when executing on the 6-stage pipeline?
- All programs will run faster.
 - All programs will run at least as fast.
 - Depending on the clock periods of the two pipelines, some programs will run at least as fast, while others may be slowed down.
 - Runtimes of all programs will remain unchanged.
 - All programs will be slowed down.

Questions 15—20 refer to the following snippet of LC2K2 code that you want to run on your 6-stage pipeline. Register R0 is 0. All other registers are initialized to 1.

```

    add 1 1 1
    lw  1 3 100
    lw  2 4 200
    add 3 3 1
    add 4 4 2
    beq 5 6 skip
skip add 1 1 2
    add 3 3 5
    add 5 5 6

```

Using the compiler SubOptima-2, you obtain the following piece of code that is guaranteed to run correctly on your 6-stage pipeline.

```

    add 1 1 1
    noop
    noop
    lw  1 3 100
    noop
    lw  2 4 200
    add 3 3 1
    noop
    add 4 4 2
    beq 5 6 skip
skip add 1 1 2
    add 3 3 5
    noop
    add 5 5 6

```

15. How many cycles does it take for the compiler-generated code to run on the 6-stage pipeline?
- a. 18
 - b. 19
 - c. 21
 - d. 22
 - e. 23
16. What are the contents of the ALUresult field in the EX2|MEM register at the end of cycle 7 in the execution of the compiler-generated program?
- a. 0
 - b. 1
 - c. 102
 - d. 201
 - e. None of the above
17. During the execution of the compiler-generated code, how many times is the EX2|MEM \rightarrow EX1 forwarding path used to provide inputs to the EX1 stage?
- a. 0
 - b. 1
 - c. 2
 - d. 3
 - e. 4

Unhappy with the performance of the SubOptima-2 compiler, you decide to take a closer look into the issue of data hazards in your 6-stage pipeline.

18. Consider the case of an R-type instruction that has a data hazard with a subsequent instruction. (For example, add 1 1 1 and lw 1 3 100.) What is the minimum number of cycles that should separate the Fetch stages of such instructions so that forwarding can correctly resolve this hazard?

- a. 1 (i.e., the two instructions can execute in consecutive cycles)
- b. 2
- c. 3
- d. 4
- e. 5

19. Consider two instructions with a **load** hazard between them. (For example, lw 1 3 100 and add 3 3 1.) What is the minimum number of cycles that should separate the Fetch stages of such instructions so that forwarding can correctly resolve the load hazard?

- a. 1 (i.e., the two instructions may execute in consecutive cycles)
- b. 2
- c. 3
- d. 4
- e. 5

20. What is the minimum number of noop's that must be inserted to the original code so that forwarding correctly resolves all data hazards? Recall that since this datapath cannot stall, the compiler is responsible for inserting all required noop's. No reordering of original instructions allowed.

- a. 1
- b. 2
- c. 3
- d. 4
- e. 5