Habib University

Dhanani School of Science and Engineering

Computer Architecture
EE 371 / CS 330 / CE 321 (Spring 2023)

Homework 3

Release Date: March 17, 2023	**Due by:** March 31, 2023 11:59 PM

Total marks: 100	**Marks obtained:**

Student Names:	**Student IDs:**	**Section:**

Purpose:

Instructions:
1. This assignment should be done in pairs.
2. You are allowed to take the help of ChatGPT to solve the design problems. If you use ChatGPT, please write down in what way you used help from ChatGPT. Please be warned that the ChatGPT may propose wrong solutions, and you will not be excused because ChatGPT misguided you.
3. All questions should be answered in **black ink only**.
4. Scan your answer sheet and upload it on HU LMS before the due date.

Grading Criteria:
1. Your assignments will be checked by instructor/TA.
2. You can also be asked to give a viva where you will be judged whether you understood the question yourself or not. If you are unable to answer correctly to the question you have attempted right, you may lose your marks.
3. Zero will be given if the assignment is found to be plagiarized.
4. Untidy work will result in reduction of your points.

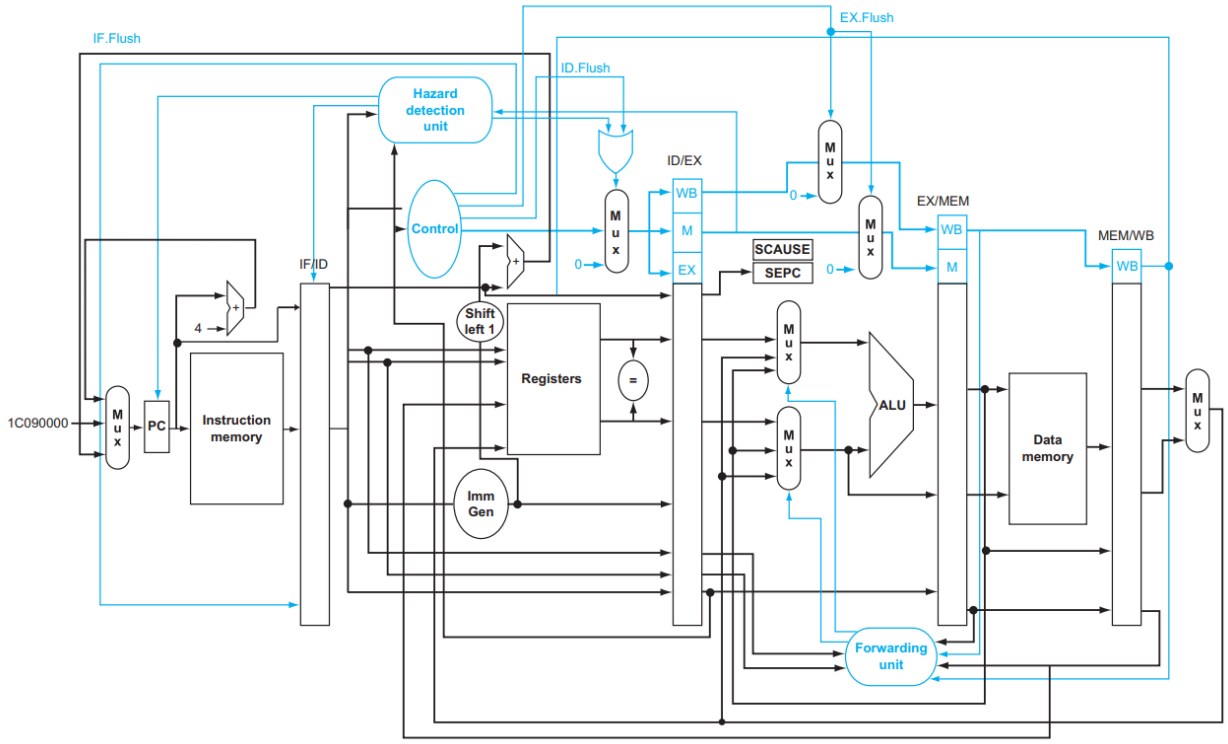
Submission policy:
1. No submission will be accepted after the instructor releases the solution on HU LMS.

CLO Assessment:
This assignment assesses students for the following course learning outcomes.

Course Learning Outcomes		CLO Assessed
CLO 1	***Explain*** the role of ISA in modern processors and instruction encodings and assembly language programming	
CLO 2	***Explain*** the architecture and working of a single cycle processor	
CLO 3	***Design*** the architecture to mitigate issues of a pipelined processor	✓
CLO 4	***Analyze the*** performance of cache operations	

Question 1: Pipeline Register Design
[5 marks]

What are the total number of bits needed for each pipeline register in the following pipelined RISC-V processor. Also break it down into the individual fields of each pipeline register and how many bits are needed for each field.



Question 2: Pipeline Basics
[5 marks]

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

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

Also, assume that instructions executed by the processor are broken down as follows:

R-type	beq	ld	sd
40%	15%	20%	25%

i. What is the clock cycle time in a pipelined and non-pipelined processor?
ii. What is the total latency of an ld instruction in a pipelined and non-pipelined processor?

iii. 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?

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

v. Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

Question 3: Pipeline Hazard Basics
[5 marks]

A hypothetical processor has 9 stages of a pipeline as shown in table below. The first row in the table below shows the pipeline stage number, second row gives the name of each stage, and third row gives the delay of each stage in Nano-seconds. The name of each stage describes the task performed by it. Each stage takes 1 cycle to execute. This processor stores all the register contents in a compressed fashion. After fetching the operands the operands are first decompressed, and before saving the results in register file, the results are first compressed.

1	2	3	4	5	6	7	8	9
Instruction Fetch	Instruction Decode	Operand Fetch	Decompress Operands	Instruction Execute 1	Instruction Execute 2	Memory Access	Compress results	Register Write back
1ns	1.4ns	1.1ns	2.2ns	2.3ns	2.3ns	1.3ns	2.2ns	1.2ns

i. How many cycles are required to implement/execute one instruction on this pipeline?

ii. How many cycles are required to execute 17 instructions on this pipeline? Assume that no stall cycles occur during the execution of all instructions.

iii. Assume that all necessary bypass circuitry is implemented in this 9 stage pipeline. How many cycles will the pipeline stall during the execution of below given two instructions?

```
a. x5 = Load from memory
b. x2 = x5 + x7
```

iv. Assume that no bypass circuitry is implemented in this 9 stage pipeline. How many cycles will the pipeline stall during the execution of above mentioned two instructions?

v. Assume that all necessary bypass circuitry is implemented in this 9 stage pipeline. How many cycles will the pipeline stall during the execution of below given two instructions?

```
a. x5 = x1 + x3
b. x2 = x5 + x7
```

vi. Assume that no bypass circuitry is implemented in this 9 stage pipeline. How many cycles will the pipeline stall during the execution of above mentioned two instructions?

vii. How much total time (in ns) is required to execute one entire instruction if the nine stages were **not pipelined**?

viii. What is the delay of 1 cycle (in ns) when all the nine stages are pipelined?

ix. What is the total time (in ns) required to execute one entire instruction if the nine stages are pipelined?

x. Below are given two set of codes. For each set of code, mention if forwarding circuit can avoid all the stalls in the code?

```
a. add     x1, x2, x3
   add     x6, x7, x8
```

```
   add     x4, x1, x5

b. ld      x1,0(x3)
   add     x2, x1, x3
```

Question 4: Pipeline Diagram
[5 marks]

Consider the following loop.

```
LOOP: ld x11, 8(x13)
      ld x10, 0(x13)
      add x12, x10, x11
      addi x13, x13, -16
      bne x12, x0, LOOP
```

Assume that perfect branch prediction is used (no stalls due to control hazards), that the pipeline has full forwarding support, and that branches are resolved in the EX (as opposed to the ID) stage.

i. Show a pipeline execution diagram for the first two iterations of this loop.
ii. Mark pipeline stages that do not perform useful work. How often while the pipeline is full do we have a cycle in which all five pipeline stages are doing useful work?

Question 5: Energy Consideration in Pipeline Design
[5 marks]

This exercise explores energy efficiency and its relationship with performance. Problems in this exercise assume the following energy consumption for activity in Instruction memory, Registers, and Data memory. You can assume that the other components of the datapath consume a negligible amount of energy. ("Register Read" and "Register Write" refer to the register file only.)

I-Mem	1 Register Read	Register Write	D-Mem Read	D-Mem Write
120pJ	80pJ	60pJ	120pJ	100pJ

Keep in mind that reading two registers (instead of one) will double the energy consumption of register reading. Assume that components in the datapath have the following latencies. You can assume that the other components of the datapath have negligible latencies.

I-Mem	Control	RegisterReadorWrite	ALU	D-MemReadorWrite
200ps	150ps	90ps	90ps	250ps

i. How much energy is spent to execute an add instruction in a single-cycle design and in the five-stage pipelined design?
ii. What is the worst-case RISC-V instruction in terms of energy consumption? What is the energy spent to execute it?
iii. If energy reduction is paramount, how would you change the pipelined design? What is the percentage reduction in the energy spent by an ld instruction after this change?
iv. What other instructions can potentially benefit from the change discussed in (iii)?

v. How do your changes from (iii) affect the performance of a pipelined CPU?
vi. We can eliminate the MemRead control signal and have the data memory be read in every cycle, i.e., we can permanently have MemRead=1. Explain why the processor still functions correctly after this change. If 25% of instructions are loads, what is the effect of this change on clock frequency and energy consumption?

Question 6: Pipeline Design Optimization for Memory Access
[5 marks]

ld is the instruction with the longest latency on the RISC-V implementation of single-cycle non-pipelined processor discussed in class. If we modified ld and sd so that there was no offset (i.e., the address to be loaded from/stored to must be calculated and placed in rs1 before calling ld/sd), then no instruction would use both the ALU and Data memory. This would allow us to reduce the clock cycle time. However, it would also increase the number of instructions, because many ld and sd instructions would need to be replaced with ld/add or sd/add combinations.

i. What would the new clock cycle time be for non-pipelined processor?
ii. In the non-pipelined version, would a program with the instruction mix presented in Question 2 run faster or slower on this new CPU? By how much? (For simplicity, assume every ld and sd instruction is replaced with a sequence of two instructions.)
iii. What is the primary factor that influences whether a program will run faster or slower on the new CPU?
iv. Do you consider the original CPU a better overall design; or do you consider the new CPU a better overall design? Why?
v. As a result of the change, the MEM and EX stages of the pipelined version of the processor can be overlapped and the pipeline has only four stages. How will the reduction in pipeline depth affect the cycle time?
vi. How might this change improve the performance of the pipeline?
vii. How might this change degrade the performance of the pipeline

Question 7: Pipeline Design for New Instructions
[50 marks]

Consider the following instructions that are not found in the RISC-V architecture:

i. Load Word Register

```
lwr rd, rs2(rs1)            // Reg[rd] = Mem[Reg[rs1] + Reg[rs2]]
```

ii. Add 3 operands

```
add3 rd, rs1, rs2, rs3      // Reg[rd] = Reg[rs1] + Reg[rs2] +
                            //           Reg[rs3]
```

iii. Add to Memory

```
addm rd, Offset(rs)         // Reg[rd] = Reg[rd] +
                            //           Mem[Offset + Reg[rs]]
```

iv. Branch Equal to Memory

```
beqm rs1, offset(rs2), rs3 // if(Reg[rs1]==Mem[Offset+Reg[rs2]])
                           //     PC = PC + Reg[rs3]
```

v. Store Word and Increment

```
swinc rs2, offset(rs1)      // Mem[Reg[rs1] + offset]= Reg[rs2],
                            // Reg[rs1] = Reg[rs1] + 4
```

We want to modify the RISC-V processor to support the above instructions. For parts (B) and (D) below, you can use a printed version of the figures in the book over which you can draw your suggested modifications (no need to draw the entire diagram from scratch).

For each of the above instructions, do the following:

A) [1 mark] Suggest if any of the existing instruction formats is a good choice to encode the new instruction. If not, then propose a new instruction format.
B) [3 marks] Modify the datapath and control signals of the single-cycle RISC-V processor (Figure 4.17 of the book) to execute the new instruction using the instruction format suggested in part (A). Use the minimal amount of additional hardware and clock cycles/control states. Remember when adding new instructions, don't break the operation of the standard ones.
C) [1 mark] Discuss the effect of the modification in part (B) on the latency of single-cycle non-pipelined CPU
D) [2 marks] Discuss if the suggested modification in part (B) should be handled by increasing/decreasing the number of pipelining stages that were discussed in class. Draw a pipelined version of the new processor similar to Figure 4.49 of the book.
E) [2 marks] Discuss if any new types of data hazards are introduced due to the new instruction? If yes, can they be mitigated through forwarding? Use a multi-cycle pipeline diagram like Figure 4.51 of the book to illustrate the new forwarding paths.
F) [1 mark] Discuss, based on the above analysis, why the new instruction was not made part of the RISC-V architecture.

Question 8: Structural Hazard Analysis
[5 marks]

Consider the fragment of RISC-V assembly below:

```
ld  x31, 32(x10)
sd  x11, 8(x10)
sub x12, x14, x13
add x15, x12, x13
beq x24, x0, label
add x31, x31, x14
```

Suppose we modify the pipeline so that it has only one memory (that handles both instructions and data). In this case, there will be a structural hazard every time a program needs to fetch an instruction during the same cycle in which another instruction accesses data.

i. Draw a pipeline diagram to show where the code above will stall.
ii. In general, is it possible to reduce the number of stalls/NOPs resulting from this structural hazard by reordering code?
iii. Must this structural hazard be handled in hardware? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? If so, explain how. If not, explain why not.
iv. Approximately how many stalls would you expect this structural hazard to generate in a typical program? Use the instruction mix from Question 2.

Question 9: Forwarding/Hazard-Detection Units Analysis
[5 marks]

Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a five-stage pipelined datapath:

```
and x16, x10, x9
ld  x29, 4(x16)
ld  x10, 0(x2)
sub x29, x29, x15
sd  x29, 0(x16)
```

i. If there is no forwarding or hazard detection, insert NOPs to ensure correct execution.
ii. Now, change and/or rearrange the code to minimize the number of NOPs needed. You can assume register x17 can be used to hold temporary values in your modified code.
iii. If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when the original code executes?
iv. If there is forwarding, for the first seven cycles during the execution of this code, specify which signals are asserted in each cycle by hazard detection and forwarding units in Figure 4.59 of the book.
v. If there is no forwarding, what new input and output signals do we need for the hazard detection unit in Figure 4.59? Using this instruction sequence as an example, explain why each signal is needed.
vi. For the new hazard detection unit from (v), specify which output signals it asserts in each of the first five cycles during the execution of this code.

Question 10: Forwarding Trade-offs
[5 marks]

Consider the pipelined RISC-V version with 5 stages discussed in class that does not handle data hazards (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary). Suppose that (after optimization) a typical n-instruction program requires an additional 0.3*n NOP instructions to correctly handle data hazards.

i. Suppose that the cycle time of this pipeline without forwarding is 200ps. Suppose also that adding forwarding hardware will reduce the number of NOPs from 0.3*n to 0.06*n, but increase the cycle time to 250ps. What is the speedup of this new pipeline compared to the one without forwarding?
ii. Different programs will require different amounts of NOPs. How many NOPs (as a percentage of code instructions) can remain in the typical program before that program runs slower on the pipeline with forwarding?
iii. Repeat (ii); however, this time let *x* represent the number of NOP instructions relative to *n*. (In (ii), *x* was equal to 0.3) Your answer will be with respect to *x*.
iv. Can a program with only .075*n NOPs (in the no-forwarding case) possibly run faster on the pipeline with forwarding? Explain why or why not.
v. At minimum, how many NOPs (as a percentage of code instructions) must a program have before it can possibly run faster on the pipeline with forwarding?

Question 11: Forwarding Logic Design Trade-offs
[5 marks]

This exercise is intended to help you understand the cost/complexity/ performance trade-offs of forwarding in a pipelined processor. Problems in this exercise refer to pipelined datapaths from Figure

4.53 of the book. These problems assume that, of all the instructions executed in a processor, the following fraction of these instructions has a particular type of read-after-write (RAW) data dependence.

EX to 1st Only	MEM to 1st Only	EX to 2nd Only	MEM to 2nd Only	EX to 1st and EX to 2nd
10%	25%	10%	15%	15%

The type of RAW data dependence is identified by the stage that produces the result (EX or MEM) and the next instruction that consumes the result (1st instruction that follows the one that produces the result, 2nd instruction that follows, or both). We assume that the register write is done in the first half of the clock cycle and that register reads are done in the second half of the cycle, so "EX to 3rd" and "MEM to 3rd" dependences are not counted because they cannot result in data hazards. We also assume that branches are resolved in the EX stage (as opposed to the ID stage), and that the CPI of the processor is 1 if there are no data hazards.

Assume the following latencies for individual pipeline stages. For the EX stage, latencies are given separately for a processor without forwarding and for a processor with different kinds of forwarding.

IF	ID	EX (no FW)	EX (full FW)	EX (FW from EX/ MEM only)	EX (FW from MEM/ WB only)	MEM	WB
150ps	120ps	140ps	180ps	150ps	150ps	150ps	120ps

i. For each RAW dependency listed above, give a sequence of at least three assembly statements that exhibits that dependency.

ii. For each RAW dependency above, how many NOPs would need to be inserted to allow your code from (i) to run correctly on a pipeline with no forwarding or hazard detection? Show where the NOPs could be inserted.

iii. Analyzing each instruction independently will over-count the number of NOPs needed to run a program on a pipeline with no forwarding or hazard detection. Write a sequence of three assembly instructions so that, when you consider each instruction in the sequence independently, the sum of the stalls is larger than the number of stalls the sequence actually needs to avoid data hazards.

iv. Assuming no other hazards, what is the CPI for the program described by the table above when run on a pipeline with no forwarding? What percent of cycles are stalls? (For simplicity, assume that all necessary cases are listed above and can be treated independently.)

v. What is the CPI if we use full forwarding (forward all results that can be forwarded)? What percent of cycles are stalls?

vi. Let us assume that we cannot afford to have three-input multiplexors that are needed for full forwarding. We have to decide if it is better to forward only from the EX/MEM pipeline register (next-cycle forwarding) or only from the MEM/WB pipeline register (two-cycle forwarding). What is the CPI for each option?

vii. For the given hazard probabilities and pipeline stage latencies, what is the speedup achieved by each type of forwarding (EX/MEM, MEM/WB, for full) as compared to a pipeline that has no forwarding?

viii. What would be the additional speedup (relative to the fastest processor from vii) be if we added "time-travel" forwarding that eliminates all data hazards? Assume that the yet-to-be-invented time-travel circuitry adds 100ps to the latency of the full-forwarding EX stage.

ix. The table of hazard types has separate entries for "EX to 1st" and "EX to 1st and EX to 2nd". Why is there no entry for "MEM to 1st and MEM to 2nd"?