Habib University

EE-371/CS-330/CE-321 Computer Architecture – Spring 2024
Instructors: Dr. Tariq Kamal, Dr. Farhan Khan, Dr. Waseem Hassan

Time = 50 minutes **Quiz 03 SOL** **Max Points: 20**

Instructions:

i. ***Smart watches, laptops, and similar electronics are strictly NOT allowed.***
ii. ***Answer sheets should contain all steps, working, explanations, and assumptions.***
iii. Attempt the quiz on clean papers with black/blue ink.
iv. Print your name and HU ID on all sheets.
v. Turn in your question paper along with your answer sheets.
vi. You are not allowed to ask/share your method or answer with your peers. The work submitted by you is solely your own work. Any violation of this will be the violation of HU Honor code and proper action will be taken as per university policy if found to be involved in such an activity.

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 [4 points]:

A notation that names the fields of the pipeline registers allows for a more precise notation of dependences. For example, "ID/EX.RegisterRs1" refers to the number of one register whose value is found in the pipeline register ID/EX; that is, the one from the first read port of the register file. The first part of the name, to the left of the period, is the name of the pipeline register; the second part is the name of the field in that register.

Using this notation, the two pairs of hazard conditions are:

- Type 1a. EX/MEM.RegisterRd = ID/EX.RegisterRs1
- Type 1b. EX/MEM.RegisterRd = ID/EX.RegisterRs2
- Type 2a. MEM/WB.RegisterRd = ID/EX.RegisterRs1
- Type 2b. MEM/WB.RegisterRd = ID/EX.RegisterRs2

Classify the **data hazards** in the following RISC-V assembly code using the above types:

```
and x2, x1, x3
xor x12, x2, x5
add x13, x16, x2
sub x14, x22, x2
sd x15, 100(x2)
```

Solution:
The data hazards are as follows:

■ The and–xor is a type 1a hazard.:
EX/MEM.RegisterRd = ID/EX.RegisterRs1 = x2

■ The and–add is a type 2b hazard:
MEM/WB.RegisterRd = ID/EX.RegisterRs2 = x2

■ The dependence on and–sub is not a hazard because the register file supplies the proper data during the ID stage of sub.

■ There is no data hazard between and and–sd because sd reads x2 the clock cycle after and writes x2.

Question 2 [6 points]:

Consider the following loop.

```
LOOP: ld x2, 16(x23)
      ld x1, 8(x23)
      add x3, x1, x2
      addi x23, x23, -24
      beq x3, 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.

a) [3 points] Show a pipeline execution diagram for the first two iterations of this loop.

b) [3 points] 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?

Solution:

a) The lines of code have been numbered and this numbering will be used in the multiple-clock-cycle pipeline diagram:

```
1: LOOP:ld x2, 16(x23)
2:      ld x1, 8(x23)
3:      add x3, x1, x2
4:      addi x23, x23, -24
5:      beq x3, x0, LOOP
```

Instr. No.	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC 10	CC 11	CC 12	CC 13	CC 14	CC 15	CC 16
1	IF	ID	EX	MEM	WB											
2		IF	ID	EX	MEM	WB										
-			Stall	Stall	Stall	Stall	Stall									
3				IF	ID	EX	MEM	WB								
4					IF	ID	EX	MEM	WB							
5						IF	ID	EX	MEM	WB						

1							IF	ID	EX	MEM	WB					
2								IF	ID	EX	MEM	WB				
-									Stall	Stall	Stall	Stall	Stall			
3										IF	ID	EX	MEM	WB		
4											IF	ID	EX	MEM	WB	
5												IF	ID	EX	MEM	WB

b) ! indicates a stage that does not do useful work.

Instr. No.	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC 10	CC 11	CC 12	CC 13	CC 14	CC 15	CC 16
1	IF	ID	EX	MEM	WB											
2		IF	ID	EX	MEM	WB										
-			Stall	Stall	Stall	Stall	Stall									
3				IF	ID	EX	MEM!	WB								
4					IF	ID	EX	MEM!	WB							
5						IF	ID	EX	MEM!	WB !						

1							IF	ID	EX	ME M	WB					
2								IF	ID	EX	ME M	WB				
-									Stall	Stall	Stall	Stall	Stall			
3										IF	ID	EX	ME M!	WB		
4											IF	ID	EX	ME M!	WB	
5												IF	ID	EX	ME M!	WB !

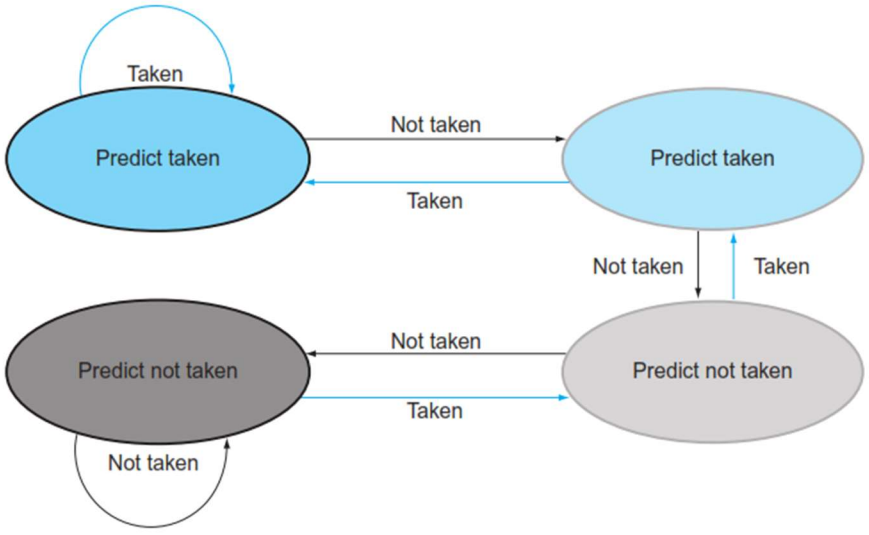
Question 3 [4 points]:

Consider the following repeating pattern of branch outcomes (e.g., in a loop):

NT, NT, NT, T, NT

a) [2 points] What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

b) [2 points] What is the accuracy of the 2-bit predictor for the first four branches in this pattern, assuming that the predictor starts off in the state "predict not taken"?



Solution:

a) The accuracy of always-taken predictor can be calculated from the following:

PREDICTION	T	T	T	T	T
ACTUAL	NT	NT	NT	T	NT
OUTCOME	Incorrect	Incorrect	Incorrect	Correct	Incorrect

Accuracy (T) = 1/5 = 0.2

The accuracy of always-not-taken predictor can be calculated from the following:

PREDICTION	NT	NT	NT	NT	NT
ACTUAL	NT	NT	NT	T	NT
OUTCOME	Correct	Correct	Correct	Incorrect	Correct

Accuracy (NT) = 4/5 = 0.8

b)

PREDICTION	NT	NT	NT	NT
ACTUAL	NT	NT	NT	T
OUTCOME	Correct	Correct	Correct	Incorrect

Accuracy (DP) = 3/4 = 0.75

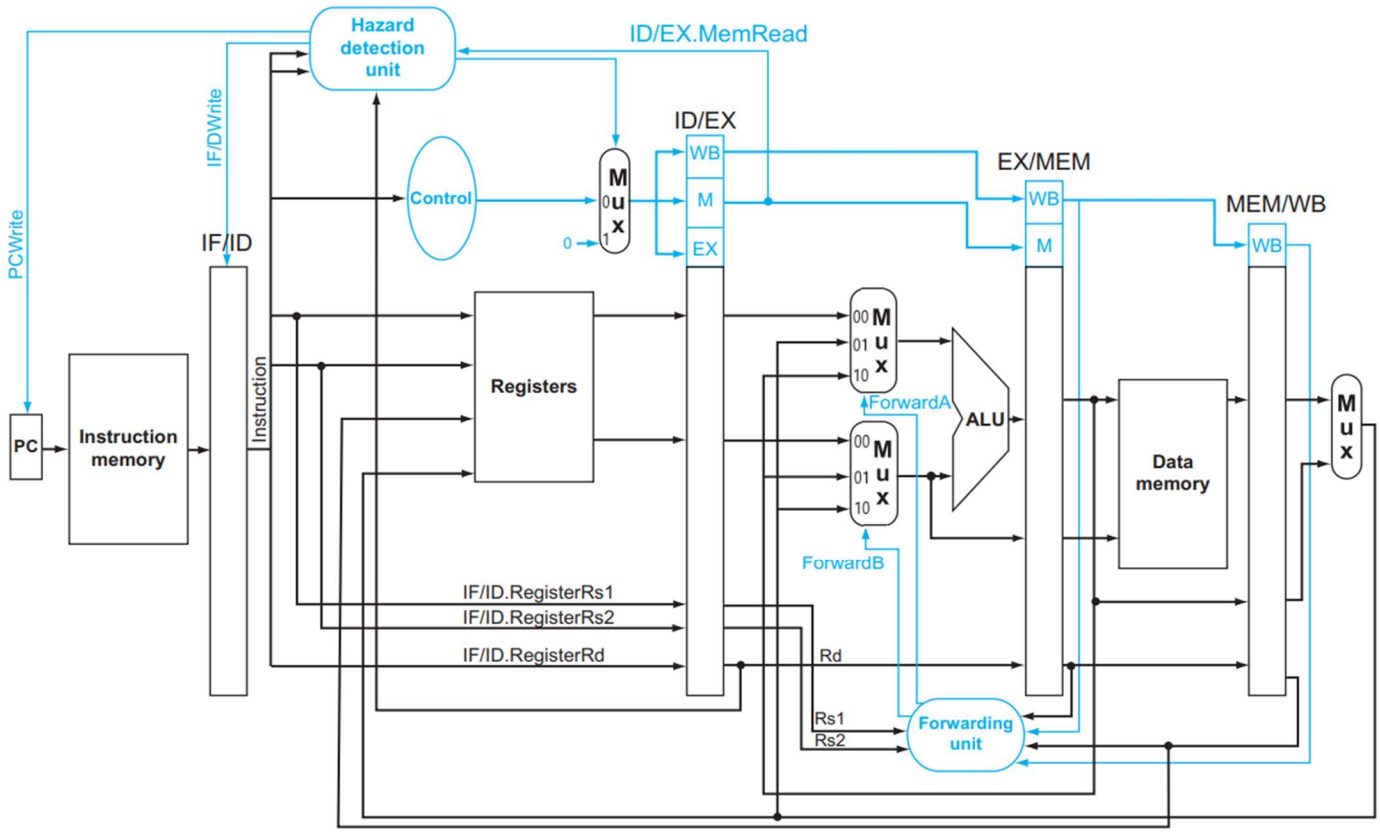
Question 4 [6 points]:

Assume that the following sequence of RISC-V assembly instructions is executed on a five-stage pipelined RISC-V processor:

```
add x18, x19, x20
ld x31, 88(x18)
ld x30, 80(x18)
sub x31, x31, x30
sd x31, 72(x18)
```

a) [3 points] If there is no forwarding or hazard detection, insert NOPs to ensure correct execution.

b) [3 points] If the processor has the forwarding unit implemented, but not the hazard detection unit, what happens when the original code executes on the processor?



Solution:

a) The lines of code have been interspersed with nops.

```
add x18, x19, x20
nop
nop
ld x31, 88(x18)
ld x30, 80(x18)
nop
nop
sub x31, x31, x30
nop
nop
sd x31, 72(x18)
```

b) A hazard detection unit is needed when a load-use data hazard occurs. All the other hazards can be catered to by the forwarding unit. Since there is a load-use data hazard between ld x30, 80(x18) and sub x31, x31, x30 i.e., for two consecutive instructions, the second instruction uses the destination register of the first one, absence of hazard detection will disrupt the execution of the given code sequence in such a way that due to forwarding, x30 will receive the computed effective address which is meant for x18 instead of the correct data destined for the x30 register.