The Operating System needs to know what two things in order to deal with an interrupt?

Predicting the direction of a branch is not enough. What else is necessary?

Artificial intelligence techniques can provide more accurate branch prediction than existing approaches - why are these techniques not used?

What are the three types of pipeline hazards?

Why do we care about techniques like forwarding, since hazards can be avoided by introducing stalls in the pipeline? Why do we want to avoid stalls?

Branch prediction requires the ability to squash (or undo) instructions. How do you "squash" an instruction in a pipelined machine? What must be prevented from happening?

Is it possible to have a WAW dependency in the MIPS 5-stage pipeline? Why or why not?

Which is on average more effective, dynamic or static branch prediction?

Briefly describe a Tournament Branch Predictor. Be sure to touch on how it works and what characteristic of branch predictors it is exploiting. Pictures are encouraged.

[4] In your first design of a 5-stage pipeline (F,D,E,M,W), F takes 26 time units, D takes 27, E takes 50, M takes 52, and W takes 51.

a) What will the clock cycle time be for this pipeline?

b) Is it a balanced pipeline? If not, explain what you could do to fix it. What would the cycle time be now?

[8] Here is a code sequence.

load R5, 0(R13)

or R1, R5, R4

add R1, R4, R2

sub R2, R4, R6

store R2, 5(R6)

Indicate all dependencies (draw lines/arrows between them, and write beside each line/arrow which hazard is involved).

[9] You are given a machine with an 8 stage pipeline, which writes to the register file during the D1 cycle and reads during D2. This machine uses neither a branch delay slot nor a load delay slot. These are the stages:

F D1 D2 E1 E2 M1 M2 W

a. Assuming this machine has a branch predictor and the branch condition is calculated by the end of the D1 stage, how big is the branch penalty (measured in cycles) when the prediction is incorrect? What if the branch condition is calculated by the end of E1?

b. Assume on a load the value is available at the end of M2. How many stall cycles stalls would this machine need on a load if it has forwarding logic and you are forwarding to E1? What if you were forwarding to E2?

c) What type of data hazard does the above pipeline need to worry about?

d) If the above pipeline were modified to support out of order completion, what new data hazard would be introduced?

e) If in addition to completing out of order, instructions were allowed to issue out of order, what new data hazard would be introduced?

[15] Here is a code sequence.

```
load    R5, 8(R9)

load    R2, 4(R6)

add     R3, R1, R2

sub     R1,R7,R6

load    R4, 0(R1)

and     R4,R9,R10
```

Assuming a standard 5-stage pipeline that does not support hazard detection and does no forwarding,

a) Indicate all dependencies (draw lines/arrows between them, and write beside each line/arrow which hazard is involved).

b) Insert as many No Operations (NOPS) as required in order to ensure this code runs correctly. (Remember, writes to the register file occur on the first half of the cycle, and reads occur during the second half).

c) Circle the NOPs that can be removed if forwarding and hazard detection logic is implemented.

d) Assuming the pipeline has been modified so that it does support hazard detection and forwarding, schedule the code to remove as many stalls as possible. How many NOPS are left?

e) If R9 contains the value 0x12, what address in memory is accessed when executing the first load instruction?

[6] In class, we talked about the cycle by cycle steps that occur on different interrupts. For example, here is what happens if there is an illegal operand interrupt generated by instruction i (note this machine has a 6 stage pipeline):

	1	2	3	4	5	6	7	8	9	
i	IF	ID	EX	M1	M2	WB	<- Interrupt detected			
i+1		IF	ID	EX	M1	M2	WB	<- Instruction Squashed		
i+2			IF	ID	EX	M1	M2	WB	<- Trap Handler fetched	
i+3				IF	ID	EX	M1	M2	WB	
i+4					IF	ID	EX	M1	M2	WB

Fill out the following table if instruction i+1 experiences a fault in the M1 stage (page fault, for example) and we are assuming **imprecise** interrupts:

	1	2	3	4	5	6	7	8	9	10	
i	IF	ID	EX	M1	M2	WB					
i+1		IF	ID	EX	M1	M2	WB				
i+2			IF	ID	EX	M1	M2	WB			
i+3				IF	ID	EX	M1	M2	WB		
i+4					IF	ID	EX	M1	M2	WB	
i+5						IF	ID	EX	M1	M2	WB

Assuming we are supporting **precise** interrupts, what happens in this case?

	1	2	3	4	5	6	7	8	9	10		
i	IF	ID	EX	M1	M2	WB						
i+1		IF	ID	EX	M1	M2	WB	<- M1 stage has Page Fault				
i+2			IF	ID	EX	M1	M2	WB	<- Illegal Instruction detected			
i+3				IF	ID	EX	M1	M2	WB	<- IF stage has page fault		
i+4					IF	ID	EX	M1	M2	WB		
i+5						IF	ID	EX	M1	M2	WB	
i+6							IF	ID	EX	M1	M2	WB
i+7								IF	ID	EX	M1	M2
