(1) [2] What is one of the simplest (and oldest) techniques for exploiting parallelism among instructions?

(2) [2] What is the primary difference between Tomasulo's algorithm and Scoreboarding?

(3) [2] Compilers have trouble doing code scheduling on code that involves reads and writes to memory. Why?

(4) [2] What are the 3 pipeline hazards? Which one can be solved by providing more resources?

(5) [2] What are the 3 types of data hazards? Which one can be solved by using forwarding?

(6) [3] Why is there a desire to create larger basic blocks? Give one example of a way to create a bigger basic block.

(7) [3] A vector processor is a type of SIMD architecture. What makes it different than a "normal" SIMD machine? (Why does it have a separate section in the SIMD chapter in the book?)

(8) [4] In your first design of a 6-stage pipeline (F,D,E1,E2,M,W), F takes 29 time units, D takes 27, E1 takes 16, E2 takes 14, M takes 31, and W takes 28.

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?

(9) [10] The standard MIPS implementation has a 5-stage pipeline, writes to the register file during the first half of the D cycle and reads during the second half. If the machine is redesigned to be a 7-stage pipeline, with the following stages:

F D E1 E2 M1 M2 WB

a) Assuming this machine has a branch predictor and the branch condition is calculated by the end of the D stage, how big is the branch penalty (measured in cycles) when the prediction is incorrect? What if the branch condition is not calculated until the end of E2?

b) How many load delay bubbles would this machine have (assuming it has forwarding logic and you are forwarding to E2) if the memory returns the value by the end of M2? M1?

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 the pipeline were modified to support out of order issue, what new data hazard would be introduced?

(10) [9] 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 7 stage pipeline and supports imprecise interrupts. RR stands for Register Read):

	1	2	3	4	5	6	7	8	9	10	11	12	13
i	IF	ID	RR	EX	M1	M2	WB	<- Interrupt detected					
i+1		IF	ID	RR	EX	M1	M2	WB	<- Instruction Squashed				
i+2			IF	ID	RR	EX	M1	M2	WB	<- Trap Handler fetched			
i+3				IF	ID	RR	EX	M1	M2	WB			

Fill out the following table if instruction i+1 experiences a fault in the EX stage:

	1	2	3	4	5	6	7	8	9	10	11	12	13
i	IF	ID	RR	EX	M1	M2	WB						
i+1		IF	ID	RR	EX	M1	M2	WB					
i+2			IF	ID	RR	EX	M1	M2	WB				
i+3				IF	ID	RR	EX	M1	M2	WB			
i+4					IF	ID	RR	EX	M1	M2	WB		
i+5						IF	ID	RR	EX	M1	M2	WB	
i+6							IF	ID	RR	EX	M1	M2	WB

Assuming precise interrupts are being supported, what happens in this case?

	1	2	3	4	5	6	7	8	9	10	11	12	13
i	IF	ID	RR	EX	M1	M2	WB	<- EX stage has fault					
i+1		IF	ID	RR	EX	M1	M2	WB	<- Inst Decode has Illegal Instruction				
i+2			IF	ID	RR	EX	M1	M2	WB				
i+3				IF	ID	RR	EX	M1	M2	WB			
i+4					IF	ID	RR	EX	M1	M2	WB		
i+5						IF	ID	RR	EX	M1	M2	WB	
i+6							IF	ID	RR	EX	M1	M2	WB
i+7								IF	ID	RR	EX	M1	M2

What is the maximum number of exceptions that could happen simultaneously in the above machine? Why?

(11) [5] The book states that slow and wide SIMD architectures can be more power efficient than fast and narrow architectures. Explain why. Also, explain the underlying assumption that is being made, and why it is that we are still making narrow fast machines.

(12) [5] What is the primary difference between superscalar and VLIW processors? Give one advantage and one disadvantage to using each approach. (These have to be different - in other words, if the advantage of superscalar is X, then you can't say a disadvantage of VLIW is that it can't do X.)

(13) [5] Supporting precise interrupts in machines that allow out of order completion is a challenge. Briefly explain what a precise interrupt is, why precise interrupt support is so important in modern machines, and describe the main technique used today to provide precise interrupts.

(14) [5] Processors have been built that were able to issue 8 instructions at a time using a fast clock. However, these processors are no longer being built - why not? Why would you choose a 3-issue machine over an 8-issue machine, if the clock rates were the same?

(15) [9] You have been writing C programs for a simple, non-pipelined machine. Your new job is to write C programs for a heavily pipelined, high performance processor. These new programs must execute as fast as possible (the emphasis is on response time, not throughput).

a) Give an example of how you will change the way you write your C program. Explain in detail why you decided to make the change (what is the problem you are overcoming?)

b) Give another example of how you will change the way you write your C program. Explain in detail why you decided to make the change (what is the problem you are overcoming?)

c) Which of your two changes is likely to make the biggest difference in performance, and why?

(16) [11] Assume there are 8 logical and 16 physical registers. On the left below is the register mapping upon entering the code sequence. Your job is to fill in the mappings after the execution of the code.

BEFORE	
Logical	Physical
0	15
1	14
2	13
3	12
4	11
5	10
6	9
7	8

AFTER	
Logical	Physical
0	
1	
2	
3	
4	
5	
6	
7	

Free Pool: 0,2,4,6,1,3,5,7

Given the following code sequence:

Before Renaming		After Renaming	
SUBF	F5,F2,F4	SUBF	P___,P13, P11
ADDF	F5,F3,F7	ADDF	P___,P___,P___
DIVF	F3,F6,F5	DIVF	P___,P___,P___
MULTF	F6,F2,F3	MULTF	P___,P___,P___

a) In the code sequence on the left (the "Before Renaming" code), draw arrows between all the dependencies and label all of the hazards.

b) Fill in the blanks in the "After Renaming" section (replace the blanks with the actual physical register names.)

c) Draw arrows between all dependencies in this renamed code and label them.

(17) [6] Suppose I have a 5-issue multithreaded machine, and there are 4 threads - A, B, C, and D. Assuming:

The number of independent instructions Thread A can find (in order): 1, then 0, then 3, then 2
The number of independent instructions Thread B can find (in order): 2, then 3, then 0, then 0
The number of independent instructions Thread C can find (in order): 1, then 1, then 2, then 2
The number of independent instructions Thread D can find (in order): 1, then 1, then 0, then 1

Fill in the following table if coarse grained scheduling is being used.

Time	Slot1	Slot2	Slot3	Slot4	Slot5
0					
1					
2					
3					

Now fill in the following table assuming the use of fine-grained scheduling.

Time	Slot1	Slot2	Slot3	Slot4	Slot5
0					
1					
2					
3					

Now, repeat the process assuming simultaneous multithreading is being used.

Time	Slot1	Slot2	Slot3	Slot4	Slot5
0					
1					
2					
3					

(18) [15] Here is a code sequence.

```
        add     R7, R8, R10

label:  lw      R2, 4(R8)

        sub     R3, R2, R4

        add     R1, R3, R8

        lw      R9, 8(R9)

        lw      R9, 4(R10)

        bne     R6, label       ; branch if R6 != 0

        add     R11, R12, R13
```

Assuming a 7-stage pipeline (F D E1 E2 M1 M2 WB) 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. Also, memory values are returned at the end of M2).

c) Circle the NOPs that can be removed if forwarding and hazard detection logic is implemented. (Values are needed at the beginning of E1 and are not available until the end of E2).

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. Assume there are no hazards through the memory system. How many NOPS are left?

Potentially useful information

F	D	E1	E2	M1	M2	W						
	F	D	E1	E2	M1	M2	W					
		F	D	E1	E2	M1	M2	W				
			F	D	E1	E2	M1	M2	W			
				F	D	E1	E2	M1	M2	W		
					F	D	E1	E2	M1	M2	W	
						F	D	E1	E2	M1	M2	W

F	D	E1	E2	M1	M2	W						
	F	D	E1	E2	M1	M2	W					
		F	D	E1	E2	M1	M2	W				
			F	D	E1	E2	M1	M2	W			
				F	D	E1	E2	M1	M2	W		
					F	D	E1	E2	M1	M2	W	
						F	D	E1	E2	M1	M2	W