Short Answer:

[3] What is the primary difference between Tomasulo's algorithm and Scoreboarding?

[2] Which data hazard occurs when instructions are allowed to complete out of order? Which one occurs when instructions are allowed to issue out of order?

[3] Why do most pipelined machines avoid the use of condition codes?

[3] What does ROB stand for, and why is it used in modern advanced pipelines? (What necessary function does it help support?)

[4] Speculation is a very useful technique for improving performance. However, it is not being used as extensively as it once was - why not?

[4] Compilers have trouble optimizing code that involves reads and writes to memory. Why? (The answer has nothing to do with how slow memory is - that is a different problem altogether).

[4] The book states that slow and wide 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.

[6] You have been writing C programs for a simple, non-pipelined machine. You have recently received a promotion, and now your 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). Give at least 2 examples of things you should do differently now, and be sure to explain in detail why (what is the problem you are overcoming?)

[6] Compare and contrast Superscalar and VLIW. Describe each, and list two advantages and one disadvantage of each approach.

[6] Briefly outline how a Vector machine works, and what type of parallelism it is exploiting.

[10] 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 DIVF instruction, including what is on the free list. (Assume that during the execution of this code, no registers are released - in other words, the free list will be shorter at the end than at the beginning.)

BEFORE	
Logical	Physical
0	2
1	4
2	6
3	8
4	10
5	12
6	14
7	0

AFTER	
Logical	Physical
0	
1	
2	
3	
4	
5	
6	
7	

Free Pool: 1,3,5,7,9,11,13,15

Free Pool:

You are given the following code sequence:

```
ADDF   F3,F4,F5
SUBF   F3,F3,F4
MULTF  F4,F2,F6
DIVF   F4,F4,F8
```

Now, rewrite the code sequence below using the actual physical register names instead of the logical ones.

```
ADDF   P__,P__,P__
SUBF   P__,P__,P__
MULTF  P__,P__,P__
DIVF   P__,P__,P__
```

[14] Here is a code sequence.

```
lw      R1, 0(R10)

lw      R2, 4(R1)

add     R3, R2, R1

sw      R3, 20(R9)

sub     R3,R7,R8

lw      R9, 4(R10)

lw      R5, 8(R10)

add     R6, R5, R4
```

a) Assuming a standard 5-stage pipeline that does not support hazard detection and does no forwarding, insert as many 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).

b) Schedule the code to remove as many stalls as possible. How many cycles does it take now?

[5] The pipelined implementation we used in class has a 5-stage pipeline, writes to the register file during the first half of the cycle and reads during the second half, and uses both a branch delay slot and a load delay slot. 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 E1 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?

[6] Find the basic blocks in this code (if there are any), and circle them.

```
label1:   lw    R2, 0(R10)
          lw    R3, 8(R1)
          add   R4, R3, R2
          sw    R4, 4(R10)
          bnez  R4, label3
label2:   sw    R4, 20(R4)
          sw    R4, 24(R0)
label3:   lw    R1, 4(R0)
          add   R5, R1, R4
          beq   R5, R6, label2
```

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

[12] Suppose I have a 5-issue multithreaded machine, and there are 3 threads - A, B, and C. Assuming:

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

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					

[8] We have talked about the cycle by cycle steps that occur on 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):

	1	2	3	4	5	6	7	8	9	10	11	12
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
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

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	<- M2 stage has page 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 at one time in the above machine? Why?