Short Answer:

(1) [2] Writes to a cache are inherently slower than reads from a cache - why?

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

(3) [2] 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) [4] Why is there a desire to create larger basic blocks? Give one example of a way to create a bigger basic block.

(5) [4] 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?

(6) [5] Intel uses a Tournament predictor in some of their processors. Describe what it is, and why it is used. Your description can (and probably should) include sketches and drawings.

(7) [5] Why is loop unrolling a valuable optimization technique? What are 2 challenges to using it?

(8) [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.

(9) [6] Supporting precise interrupts in machines that allow out of order completion is a challenge. Briefly explain what a precise interrupt is, why it is a challenge in OOO machines, and describe the main technique used today to provide precise interrupts.

(10) [5] Briefly outline how a Vector machine works, and what type of parallelism it is exploiting.

(11) [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.)

(12) [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?

(13) [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
MULTF  F5,F2,F6
SUBF   F3,F3,F5
DIVF   F7,F6,F3
```

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

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

(14) [16] Here is a code sequence.

```
label: lw    R2, 4(R1)

       add   R3, R1, R2

       sub   R3,R3,R8

       lw    R5, 8(R9)

       lw    R9, 4(R10)

       bne   R3, label      ; branch if R3 != 0

       lw    R11, 100(R12)
```

Assuming a 6-stage pipeline (F D E 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, values return from memory at the end of M2).

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

(15) [6] 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): 1, then 4, then 0, then 2
The number of independent instructions Thread B can find (in order): 3, then 0, then 2, then 1
The number of independent instructions Thread C can find (in order): 1, then 1, then 3, then 2

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					

(16) [3] Assuming a 19-bit address and a 256-byte Direct Mapped cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

(17) [3] Assuming an 19-bit address and an 80-byte 10-way Set Associative cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

(18) [2] Assuming an 19-bit address and a 328-byte Fully Associative cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

(19) [3] Given a 1 Megabyte physical memory, a 22 bit Virtual address, and a page size of 1K bytes, write down the number of entries in the Page Table, and the width of each entry.

(20) [4] Given a 1 Megabyte physical memory, a 34 bit Virtual address, and a page size of 2K bytes, write down the number of entries in the Page Table, and the width of each entry. Is there a problem with this configuration? If so, what is the problem?