Short Answer:

(1) [2] Give 1 technique that can be used to reduce the Miss Rate.

(2) [2] Give 1 technique that can be used to reduce the Hit Time. (Must be different than the technique used for the previous answer).

(3) [2] What is the primary difference between Tomasulo's algorithm and Scoreboarding?

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

(6) [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).

(7) [6] What is a Tournament predictor? Describe what it is, how it works, and why it is used. Your description can (and probably should) include sketches and drawings.

(8) [6] 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] 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?)

(10) [6] 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 an 8-stage pipeline, with the following stages:

F D RR 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?

(11) [6] Briefly outline how a Vector machine works, and what type of parallelism it is exploiting.

(12) [6] What is the primary difference between superscalar and VLIW processors? Give one advantage and one disadvantage to using each approach.

[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  F4,F2,F6
SUBF   F3,F3,F4
DIVF   F5,F4,F7
```

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__
```

(13) [16] Here is a code sequence.

lw R2, 4(R1)

add R3, R1, R2

sw R3, 20(R9)

sub R3,R7,R8

lw R5, 8(R10)

lw R9, 4(R10)

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?

(14) [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 0, then 2, then 3
The number of independent instructions Thread B can find (in order): 2, then 4, then 0, then 1
The number of independent instructions Thread C can find (in order): 2, then 1, then 3, 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					

(15) [3] Assuming an 18-bit address and a 256-byte Direct Mapped cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

(16) [3] Assuming an 18-bit address and an 80-byte 10-way Set Associative cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

(17) [2] Assuming an 18-bit address and a 324-byte Fully Associative cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

(18) [2] Given a 1 Megabyte physical memory, a 22 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.

(19) [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, how can you fix it?