(1) [2] What principle makes Virtual Memory possible?

(2) [2] Does an average program's locality behavior remain the same during the entire run of the program?

(3) [2] Write down the average memory access time equation.

(4) [3] What does TLB stand for? What does it do?

(5) [3] Writing to a cache is inherently slower than reading from a cache. Why?

(6) [4] What does ROB stand for, and why is it used in modern advanced pipelines? (What necessary function does it help support?)

(7) [4] Speculation is a very useful technique for improving performance. However, it is not being used as extensively as it once was - why not?

(8) [4] There are some hardware structures which use true LRU, even over a large number of addresses - what is an example, and why would designers do that?

(9) [6] 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.)

(10) [5] The designer has the choice of using a physically addressed cache or a virtually addressed cache. Explain the difference (drawing a picture is fine!), and give 1 advantage for each.

(11) [12] You have been writing high level language programs for a high performance, *non-pipelined* machine. You have recently received a promotion, and now your job is to write high level language programs for a heavily pipelined, high performance processor with an advanced tournament-style branch predictor and a 2 billion entry RAS. Your programs must execute as fast as possible (the emphasis is on response time), and your code will be compiled using a highly optimizing compiler on the -O3 setting.

a) Give an example of how you will change the way you write your high level language 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 high level language program. Explain in detail why you decided to make the change (what is the problem you are overcoming?) Your answer must address a different problem than your answer for part a) above.

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

(12) [15] For each of the following techniques, circle the arrow(s) associated with each of the terms in the Average Memory Access Time which indicates how (on average) that term is affected. If there is more than one answer, then circle more than one term. For example, if the HT goes up, you would circle the up arrow - if the HT goes down, circle the down arrow. If the HT is unaffected, do not circle anything. Assume there is a single L1 cache.

Increasing Associativity	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Decreasing cache size	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Decreasing Line/Block size	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Hardware Prefetching (Assume prefetched data does not arrive before needed)	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Hardware Prefetching (Assume prefetched data arrives before needed)	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Compiler optimizations (Excluding prefetching)	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Nonblocking cache	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Adding an L2 cache (Affect on L1 parameters)	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Physically Addressed Cache	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Adding a Victim Cache (Assume it is accessed in parallel)	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓
Adding a Victim Cache (Assume it is accessed in serial, not in parallel)	HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓

---

(13) [2] Circle the type of cache miss that can be changed by altering the mapping scheme.

Compulsory Conflict Capacity Coherence

(14) [3] Given the following 22-bit address and a 1024-byte Direct Mapped cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

0 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0

(15) [4] Assuming a 22-bit address and a 160-byte 5-way Set Associative cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

0 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0

(16) [2] Assuming a 22-bit address and a 320-byte Fully Associative cache with a linesize=16, show how an address is partitioned/interpreted by the cache.

0 1 1 1 0 1 1 0 1 1 1 0 0 0 1 0 1 0 0 1 1 0

(17) [5] Given a 1 Megabyte physical memory, a 22 bit Virtual address, and a page size of 512 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?

(18) [5] Given a 4 Megabyte physical memory, a 32 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. Is there a problem with this configuration? If so, how can you fix it?

(19) [6] Assume an 8-bit processor, with a 24-byte 3-way set associative cache and a linesize of 2.
This is the contents of the cache:

Set 0		
Tag	Data (0)	Data (1)
10110	0x3b	0xC1
11001	0x9D	0x90
00110	0x9F	0xA8

Set 1		
Tag	Data (0)	Data (1)
00010	0xd8	0xA9
00001	0xD4	0x9B
11011	0x2A	0xF6

Set 2		
Tag	Data (0)	Data (1)
11110	0x72	0x9A
11011	0xC4	0x25
00111	0x69	0x83

Set 3		
Tag	Data (0)	Data (1)
11111	0x4A	0xF7
01011	0x94	0xFA
00100	0x3C	0xD8

a) If the processor issues the address

1 0 1 1 0 1 1 0

Is this a hit in the cache? (YES NO) If YES, circle the entry and the data value that is returned.

b) If the processor issues the address

1 1 0 1 1 0 1 1

Is this a hit in the cache? (YES NO) If YES, circle the entry and the data value that is returned.

---

(20) [2] Circle the type of cache miss that can be reduced by using longer lines.

Compulsory Conflict Capacity Coherence

(21) [9] 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	8
1	9
2	10
3	11
4	12
5	13
6	14
7	15

AFTER	
Logical	Physical
0	
1	
2	
3	
4	
5	
6	
7	

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

Given the following code sequence:

Before Renaming		After Renaming	
SUBF	F5,F2,F4	SUBF	P___,P10,P___
ADDF	F5,F3,F5	ADDF	P___,P11,P___
DIVF	F3,F2,F7	DIVF	P___,P10,P___
MULTF	F6,F2,F3	MULTF	P___,P10,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.