Very short answer questions. You must use 10 or fewer words. "True" and "False" are considered very short answers.

(1) **[1]** Predicting the direction of a branch is not enough. What else is necessary?

(2) **[1]** Which is on average more effective, dynamic or static branch prediction?

(3) **[1]** Does an average program's locality behavior remain the same during the entire run of the program?

(4) **[1]** Is peak performance usually the same as sustained performance?

(5) **[1]** Which type of cache miss can be reduced by using shorter lines?

(6) **[1]** Which type of cache miss can be reduced by using longer lines?

(7) **[1]** Using a different mapping scheme will reduce which type of cache miss?

(8) **[1]** What pipeline hazard can be avoided by "throwing money at the problem"?

(9) **[1]** What pipeline hazard can be avoided using a technique known as value prediction?

(10) **[1]** Give 1 advantage to using a VLIW.

(11) **[1]** Give 1 disadvantage to using a VLIW.

(12) **[1]** Give 1 advantage to using a superscalar. Your answer must be different than your VLIW answer.

(13) **[1]** Give 1 disadvantage to using a superscalar. Your answer must be different than your VLIW answer.

(14) **[1]** As transistors and wires shrink, what happens to the power density?

(15) **[2]** There are two main ways to define performance - what are they?

(16) **[2]** There are two major challenges to obtaining a substantial decrease in response time when using the MIMD approach. What are they?

(17) **[2]** If I add processors but keep the job size the same, am I measuring strong or weak scaling? Does this correspond most closely to response time or throughput?

(18) **[2]** Give a one-word definition of coherence, and a one-word definition of consistency.

Short Answers (20 or fewer words)

(19) **[2]** What is a benchmark program?

(20) **[2]** Do benchmark programs remain valid indefinitely? Why or why not?

(21) **[3]** Why is it difficult/impossible to create a benchmark that will work across all classes of parallel processors?

(22) **[3]** Over the years, clock rates grew by a factor of 1000 while power consumed only increased by a factor of 30. How was this accomplished without melting the chip?

(23) **[3]** Which is harder to write a program for, a shared memory machine or a message passing machine? Why?

(24) **[3]** Which is more expensive to build - a shared memory machine, or a message passing machine? Why?

(25) **[3]** What is "leakage" current? If Vdd is lowered, what happens to the amount of leakage current, and why?

(26) **[3]** Why is it so difficult for the processing elements on a CMOS-based chip to communicate with things that located off the chip?

(27) **[4]** Speculation is a very useful technique for improving performance. However, it is not being used as extensively as it once was - why not?

(28) **[3]** What is the definition of a basic block? Why is there a desire to create a bigger one?

(29) **[2]** What is the definition of a precise interrupt?

(30) **[2]** Why is it important to support precise interrupts in modern pipelined processors?

(31) **[6]** 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.

(32) **[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?

(33) **[4]** Vector machines are an example of a SIMD style of parallel processing. They feature instructions that look like **VR0 = VR1 + VR2**. Explain briefly why these machines are able to fetch and decode many fewer instructions than a tradition processor does. Use pictures if that will help get your point across.

(34) **[6]** 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.

(35) **[4]** An important program spends 30% of its time doing memory operations (loads and stores). By redesigning the memory hierarchy you can make the memory operations 80% faster (take 20% as long), or you can redesign the hardware to make the rest of the machine 30% faster (take 70% as long). Which should you do and why? (You must show your work to get full credit.)

(36) **[6]** You are responsible for designing a new embedded processor, and for a variety of reasons you **must** use a fixed 23 bit instruction size and you **must** support at least 32 different opcodes. You would like to use a 3-operand instruction format, and have 128 registers. If it is possible to do this, draw what an instruction would look like. If it is not possible explain why, and give at least 2 different ways to solve the problem.

(37) **[4]** Find the Average Memory Access Time (AMAT) for a processor with a 1 ns clock cycle time, a miss penalty of 20 clock cycles, a miss rate of 0.10 misses per instruction, and a cache access time (including hit detection) of 1 clock cycle. Assume that the read and write miss penalties are the same and ignore other write stalls.

(38) **[12]** 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 cache size	(HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓)
Decreasing Associativity	(HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓)
Decreasing Line/Block size	(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 not accessed in parallel)	(HT: ↑ ↓	MR: ↑ ↓	MP: ↑ ↓)

---

(39) **[5]** In your first design of a 5-stage pipeline (F/D,E,M1,M2,W) F/D takes 50 time units, E takes 26, M1 takes 13, M2 takes 14, and W takes 26.

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 make it more balanced. What would the cycle time be now?

(40) **[9]** Understanding the hardware can influence how you write high-level programs. You have been writing C programs for a high performance, *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 with an advanced tournament-style branch predictor. Your programs must execute as fast as possible (the emphasis is on response time, not throughput), and your code will be compiled using a highly optimizing compiler on the -O3 setting. Loops will be unrolling by the compiler, so manually unrolling loops provides no benefit.

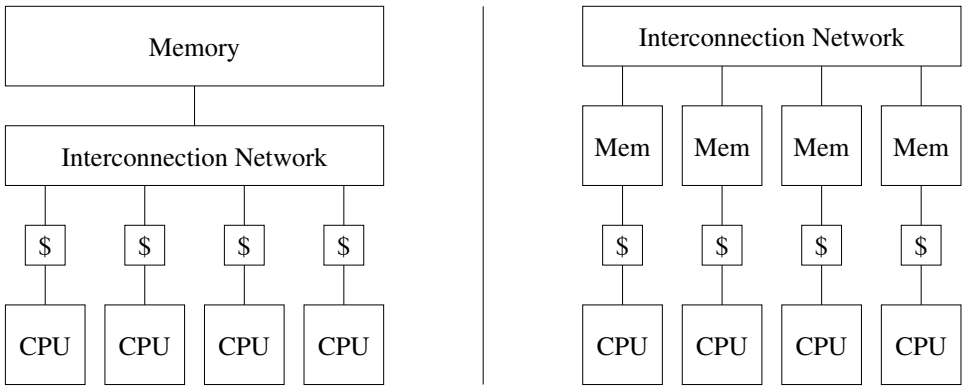
a) Give an example of how you will change the way you write your C program. Explain in detail why you decided to make the change (what is the problem you are overcoming?)

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?

---

(41) **[2]** Look at the figure below, then write below each machine if it is more likely to be a message passing or a shared memory design.



Design Approach A

Design Approach B

(42) **[7]** The MIPS 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:

F1 F2 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 slots would this machine need (assuming it has forwarding logic and you are forwarding to E2) assuming 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 in addition to completing out of order, instructions were allowed to issue out of order, what new data hazard would be introduced?

---

(43) **[2]** What is the main difference between a commodity cluster and a custom cluster?

(44) **[2]** How do two processes communicate when running on a shared memory machine?

(45) **[6]** Here is a code sequence. This is (obviously) generic code.

```
        INSTRUCTION 1

        NOP

        INSTRUCTION 2

        INSTRUCTION 3

        INSTRUCTION 4

Label:  INSTRUCTION 5

        NOP

        NOP

        INSTRUCTION 6

        INSTRUCTION 7

        NOP

        INSTRUCTION 8

        breq    CONDITION,Label         ; branch to Label if CONDITION
```

Assume there are the following dependencies in this code:

RAW between "INSTRUCTION 1" and "INSTRUCTION 2"
RAW between "INSTRUCTION 5" and "INSTRUCTION 6"
RAW between "INSTRUCTION 7" and "INSTRUCTION 8"

WAW between "INSTRUCTION 2" and "INSTRUCTION 4"
WAR between "INSTRUCTION 3" and "INSTRUCTION 4"

Draw in the arrows, and then indicate how you would schedule the code to remove the maximum number of stalls. How many are left when you are done?

(46) **[12]** Here is a code sequence.

```
sub     R3, R7, R1

add     R2, R3, R4

store   R1, 50(R5)

load    R4, 100(R7)

and     R6, R4, R7

or      R6, R9, R5
```

Assuming a standard 5-stage pipeline 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).

c) Circle the NOPs that can be removed if forwarding and hazard detection logic is implemented.

d) If there are any NOPs remaining, is it possible to reschedule the code to make them go away? If so, show how.

e) There are things the static scheduler does not know which makes guaranteeing correctness difficult. What if, at execution time, R5=200 and R7=150? Will your dependency graph and ability to schedule this code change? If so, explain/show how.

(47) **[12]** Here is a code sequence.

add R1, R2, R3

sub R11, R5, R1

xor R11, R12, R13

load R4, 0(R7)

add R6, R4, R9

and R9, R3, R5

Assuming a 6-stage pipeline (F,D,E1,E2,M,WB) that does not support hazard detection, does no forwarding, and does not use a branch delay slot:

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).

c) Circle the NOPs that can be removed if forwarding and hazard detection logic is implemented. Assume data from a load returns at the end of M, data is valid at the end of E2 and needed at the beginning of E1.

d) If there are any NOPs remaining, is it possible to reschedule the code to make them go away? If so, show how.

F	D	E1	E2	M	W				
	F	D	E1	E2	M	W			
		F	D	E1	E2	M	W		
			F	D	E1	E2	M	W	
				F	D	E1	E2	M	W
					F	D	E1	E2	M	W

(48) **[11]** Assume our machine has 8 logical and 16 physical registers. On the left below is the code you are dealing with. On the right below is the register mapping upon entering the code sequence.

load R6, 0(R4)

add R6, R6, R4

xor R4, R3, R6

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

Free Pool: 2,3,4,5,6,7,8,9

(a) Indicate all the dependencies on the code segment above.

(b) Rewrite the code sequence below using the actual physical register names instead of the logical ones.

load P___, 0(P14)

add P___, P___, P___

xor P___, P___, P___

(c) Now indicate all the dependencies in the above renamed code.

---

(49) **[2]** Superscalar and VLIW are 2 approaches used to increase ILP. What is the primary difference between them?

(50) **[6]** Suppose I have a 3-issue multithreaded machine, and there are 3 threads - A, B, and C. Assume:

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

Fill in the following table assuming fine-grained scheduling is being used.

Time	Slot1	Slot2	Slot3
0			
1			
2			
3			

Fill in the following table assuming the use of coarse-grained scheduling.

Time	Slot 1	Slot2	Slot3
0			
1			
2			
3			

Now, repeat the process assuming the use of simultaneous multithreading on a 5-issue multithreaded machine with 4 threads - A, B, C and D. Assume:

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

Time	Slot1	Slot2	Slot3	Slot4	Slot 5
0					
1					
2					
3					

---

(51) **[2]** When we talk about the number of operands in an instruction (a 1-operand or a 2-operand instruction, for example), what do we mean?

(52) **[3]** Assuming a 20-bit address and a 512-byte Direct Mapped cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

(53) **[3]** Assuming a 20-bit address and a 48-byte 6-way Set Associative cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

(54) **[2]** Assuming a 20-bit address and a 792-byte Fully Associative cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

(55) **[3]** Given an 8 Megabyte physical memory, a 26 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?

(56) **[4]** Given a 2 Gigabyte physical memory, a 46 bit Virtual address, and a page size of 16K 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?

(57) **[6]** 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
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		
i+4					IF	ID	RR	EX	M1	M2	WB	

Fill out the following table if for the same machine, instruction i experiences a fault in the EX stage (overflow, for example):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
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	
i+7								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	14
i	IF	ID	RR	EX	M1	M2	WB			<- EX stage has page fault				
i+1		IF	ID	RR	EX	M1	M2	WB		<- ID stage has arithmetic fault				
i+2			IF	ID	RR	EX	M1	M2	WB	<- IF stage has page fault				
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	WB

What is the maximum number of exceptions that could happen at a single time in the above machine (assuming no hardware errors)? Explain how you got your answer.