Very Short Answer:

(1) (2) If you are mainly worried about program size, what type of instruction set would you use? What if performance was your primary concern?

(2) (1) Do benchmarks remain valid indefinitely?

(3) (2) Issuing multiple instructions per cycle puts tremendous pressure on what two parts of the machine?

(4) (2) In class we mentioned VLIW and Superscalar as two ways to circumvent the Flynn Limit of 1. We also talked about two other approaches - what were they?

(5) (1) Peak performance does or does not track observed performance.

(6) (2) Decoupled architectures split a program into two streams. What are they?

(7) (3) Write down the 3-term CPU performance equation developed in class.

(8) (2) What is Amdahl's law (in words)?

Short Answers:

(10) (3) Why are there multiple dies per silicon wafer? Why not just fabricate one huge die per wafer?

(11) (4) What is the primary difference between Scoreboarding and Tomasulo's algorithm?

(12) (6)What is a predicated instruction? What are the advantages to using predicated instructions? When would you *not* want to use one?

(13) (6) What is the definition of a basic block? Why is there a desire to create larger ones?

(14) (6) Understanding the hardware can influence how you write programs. Give at least 2 examples of how you might write software differently for a heavily pipelined machine verses a non-pipelined one.

(15) (10) Supporting precise interrupts in machines that allow out of order completion is a challenge. Briefly explain why, and give three different techniques that can be used to provide precise interrupts.

(16) (10) Why is branch prediction important? What performance enhancing techniques have made it so? List 3 examples of existing Branch Prediction strategies in order of (average) increasing effectiveness.

(17) (10) Compare and contrast Superscalar and VLIW. Describe each, and list the advantages and disadvantages of each approach.

(14) You are given the following code sequence:

```
ADDF    F1,F1,F4
SUBF    F2,F2,F1
MULTF           F2,F2,F3
DIVF    F1,F3,F1
```

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

Free Pool: 0,2,4,9,10,13,14,15

AFTER	
Logical	Physical
0	
1	
2	
3	
4	
5	
6	
7	

Free Pool:

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__
```

(18) (16) Fill in this table for time 13. Assume that you write to the register file during the first half of the clock and read from it during the second.

Time = 9; Mult Given Priority on Read

Instruction Status

Instruction		Issue Inst	Read Operands	Execute Inst	Write Result
MULTF	F0,F2,F4	x	**X**		
SUBF	F8,F6,F2	x			
DIVF	F10,F0,F6	x			
ADDF	F6,F8,F2				

Functional Unit Status

FU	Name	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
1	Int/LS	No								
2	Mult1	Yes	Mult	F0	F2	F4				
3	Mult2									
4	Add	Yes	Sub	F8	F6	F2		1	Yes	Yes
5	Div	Yes	Div	F10	F0	F6	2		No	Yes

Register Result Status

	F0	F2	F4	F6	F8	F10	12	...	F30
FU#	2				4	5			

Time = 11,12

Instruction Status

Instruction		Issue Inst	Read Operands	Execute Inst	Write Result
MULTF	F0,F2,F4	x	x		
SUBF	F8,F6,F2	x	x		
DIVF	F10,F0,F6	x			
ADDF	F6,F8,F2				

Functional Unit Status

FU	Name	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
1	Int/LS	No								
2	Mult1	Yes	Mult	F0	F2	F4				
3	Mult2									
4	Add	Yes	Sub	F8	F6	F2				
5	Div	Yes	Div	F10	F0	F6	2		No	Yes

Register Result Status

	F0	F2	F4	F6	F8	F10	12	...	F30
FU#	2				4	5			

Time = 10; Sub Reads F2

Instruction Status

Instruction		Issue Inst	Read Operands	Execute Inst	Write Result
MULTF	F0,F2,F4	x	x		
SUBF	F8,F6,F2	x	**X**		
DIVF	F10,F0,F6	x			
ADDF	F6,F8,F2				

Functional Unit Status

FU	Name	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
1	Int/LS	No								
2	Mult1	Yes	Mult	F0	F2	F4				
3	Mult2									
4	Add	Yes	Sub	F8	F6	F2				
5	Div	Yes	Div	F10	F0	F6	2		No	Yes

Register Result Status

	F0	F2	F4	F6	F8	F10	12	...	F30
FU#	2				4	5			

Time = 13

Instruction Status

Instruction		Issue Inst	Read Operands	Execute Inst	Write Result
MULTF	F0,F2,F4	x	x		
SUBF	F8,F6,F2	x	x	**X**	**X**
DIVF	F10,F0,F6	x			
ADDF	F6,F8,F2				

Functional Unit Status

FU	Name	Busy	Fm	Fi	Fj	Fk	Qj	Qk	Rj	Rk
1	Int/LS									
2	Mult1									
3	Mult2									
4	Add									
5	Div									

Register Result Status

	F0	F2	F4	F6	F8	F10	12	...	F30
FU#									