

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

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

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

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

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

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

[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?

[2] If we make transistors and wires smaller and smaller but make no other changes, what happens to the power density?

[2] There are two main ways to define performance - what are they?

[2] What is the main difference between a commodity cluster and a custom cluster?

[2] The Operating System needs to know what two things in order to deal with an interrupt?

Short Answers (25 or fewer words)

[2] Area on the die used to be the most critical design constraint. That is no longer true - what is the most critical factor now? Why?

[2] Clock rates have grown by a factor of 1000 while power consumed has only grown by a factor of 30. How was this accomplished, and why did it work?

[2] What is a benchmark program?

[2] Do benchmark programs remain valid indefinitely? Why or why not?

[2] Why is it difficult to come up with good benchmarks that will work across all parallel processors?

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

[2] Which is easier to write a program for, a shared memory machine or a message passing machine? Why?

[2] Which is more expensive to build - a shared memory machine, or a message passing machine? Why?

[2] What is the definition of a basic block? Why is there a desire to create a bigger one?

[2] What are the two biggest challenges to obtaining a substantial decrease in response time when using a MIMD parallel processor?

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

[2] Supporting precise interrupts in a machine that allows out of order completion is a challenge. What is a precise interrupt, and why is it important to support them?

[4] What is the primary difference between superscalar and VLIW processors? Give two advantages to using a superscalar processor, and 1 advantage to using a VLIW.

[3] The designer has the choice of using a physically addressed cache or a virtually addressed cache. Explain the difference, and give 1 advantage for each.

[4] 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?) (Note - make sure you are describing things that you can do in a high level language, not things that are done at the assembly language level.)

[9] For each technique in the following table, indicate how (on average) the 3 terms of the AMAT equation are affected (assume there is a single L1 cache). The first one is done as an example. (Decreases = "-" or "V"; Increases = "+" or "^"; leave blank if unaffected).

| Technique | Hit Time | Miss Rate | Miss Penalty |
|---|----------|-----------|--------------|
| Increasing Line/Block Size | | - | + |
| Decreasing Cache Size | | | |
| Increasing Associativity | | | |
| Compiler Optimizations (excluding prefetch) | | | |
| Hardware Prefetching | | | |
| Virtually Addressed Cache | | | |
| Non-blocking Cache | | | |

[4] In your first design of a 5-stage pipeline (F,D,E,M,W) F takes 24 time units, D takes 26, E takes 29, M takes 50 and W takes 19.

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?

[2] An important program spends 70% of its time doing Integer operations, and 30% of its time doing floating point arithmetic. By redesigning the hardware you can make the Integer unit 40% faster (take 60% as long), or you can make the Floating Point unit 95% faster (take 5% as long). Which should you do and why? (You must show your work to get full credit.)

[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:

F D E1 E2 M1 M2 M3 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?

b) How many load delay slots would this machine need (assuming it has forwarding logic and you are forwarding to E1) assuming the memory returns the value by the end of M3? M2?

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?

[2] You are responsible for designing a new embedded processor, and for a variety of reasons you must use a fixed 20 bit instruction size. You would like to support 32 different opcodes, use a 3-operand instruction format, and have 64 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.

[2] Assuming a 24-bit address and a 2k-byte Direct Mapped cache with a linesize=2, show how an address is partitioned/interpreted by the cache.

[3] Assuming a 24-bit address and a 96-byte 3-way SA cache with a linesize=4, show how an address is partitioned/interpreted by the cache.

[2] Assuming a 24-bit address and a 248-byte FA cache with a linesize=8, show how an address is partitioned/interpreted by the cache.

[3] Given a 512 Megabyte physical memory, a 40 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?

[2] The CPI of the Derpster Megatron is 2, assuming a perfect memory system (all references to memory take a single cycle.) Assume we are able to keep a perfect instruction memory system but must use a more realistic data memory system - a data cache with a miss rate of 20% and a miss penalty of 50 cycles. If 30% of all instructions are loads or stores, what does the CPI become in this case?

In this question, we are going to wire up a 12-bit processor. The machine is word-addressable, where a word is 12 bits. immediates are sign extended, Offset is not. The machine has 3 different instruction formats: R, I, and J. Memory takes a single cycle to return a value.

| | | | | |
|---------|--|-----|-----------|-------|
| R-type: | <i>(Arithmetic and logical: $rd = rs1 \text{ OP } rs2$)</i> | | | |
| Opcode | rd | rs1 | rs2 | funct |
| 11-8 | 7-6 | 5-4 | 3-2 | 1-0 |
| I-type: | <i>(Arithmetic and logical: $rd = rs1 \text{ OP } Immediate$)</i> | | | |
| | <i>(Load: $rd = mem[rs1+Immediate]$)</i> | | | |
| | <i>(Store: $mem[rs1+Immediate]=rd$)</i> | | | |
| | <i>(Branch: $if (rd \text{ OP } rs1)=COND, PC=PC+Immediate$)</i> | | | |
| Opcode | rd | rs1 | Immediate | |
| 11-8 | 7-6 | 5-4 | 3-0 | |
| J-type: | <i>(Jump: $PC = Offset$)</i> | | | |
| Opcode | Offset | | | |
| 11-8 | 7-0 | | | |

The ALU can perform 4 functions, written this way: OP [ALU1 ALU0]
 Add [11], XOR [10], AND [01], OR [00]

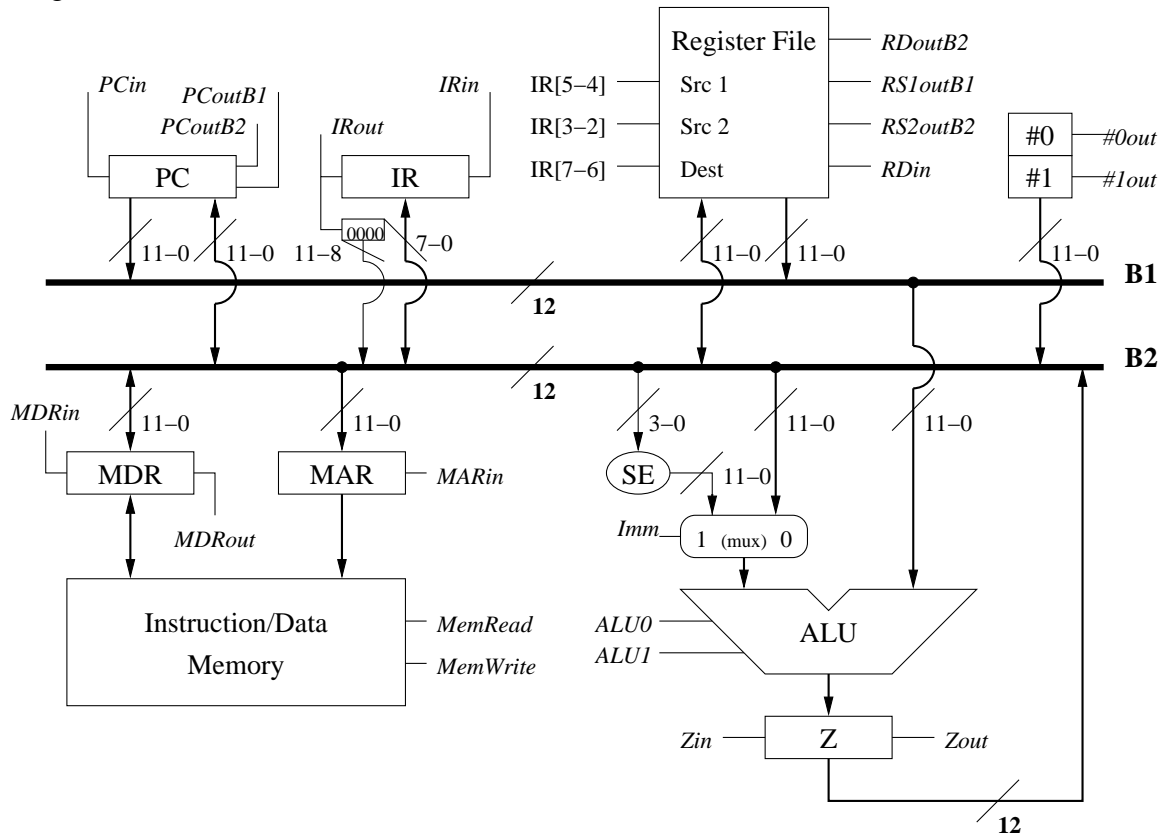
Here are some of the instructions that have been defined:

| Name | Opcode(Funct) | Name | Opcode(Funct) | Name | Opcode(Funct) |
|------|---------------|---------|---------------|---------|---------------|
| NOP | 0000(00) | lw | 0001(xx) | sw | 0011(xx) |
| NOT | 1000(00) | BEQZ | 0100(xx) | j | 0101(xx) |
| AND | 1000(01) | AND Imm | 1001(xx) | ADD | 1100(01) |
| OR | 1000(10) | OR Imm | 1010(xx) | ADD Imm | 1101(xx) |
| XOR | 1000(11) | XOR Imm | 1011(xx) | SUB | 1100(01) |

Here are the control signals.

| | | | | | | |
|----------|---------|----------|--------|-------|---------|----------|
| PCin | IRin | MDRin | MARin | Zin | RDin | Imm |
| PCoutB1 | PCoutB2 | IRout | MDRout | Zout | RDoutB2 | RS1outB1 |
| RS2outB2 | MemRead | MemWrite | #0out | #1out | ALU0 | ALU1 |

Here is a diagram of the machine.



[6] Fill in the microcode steps necessary to perform an instruction fetch (incrementing the PC is considered part of fetch).

| Step | PCin | IRin | MARin | MDRin | Zin | IRout | PCoutB1 | PCoutB2 | Mout | Zout | RoutB2 | RoutB1 | RoutB2 | #0out | #1out | ALU1 | ALU0 | MemRead | MemWrite | Imm |
|------|------|------|-------|-------|-----|-------|---------|---------|------|------|--------|--------|--------|-------|-------|------|------|---------|----------|-----|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |

[2] Write down the binary bit pattern (the bit pattern that will be in the IR after you have done the instruction fetch) for the instruction: **ADD R3, R2, -1**

[6] Now that you have done the instruction fetch, fill in the microcode steps necessary to perform the following instruction: **XOR R3, R2, -1**

| S t e p | P C i n | I R i n | M A R i n | M D R i n | Z i n | I R o u t | P C o u t B 1 | P C o u t B 2 | M D R o u t | Z o u t | R D o u t B 2 | R S o u t B 1 | R S o u t B 2 | # 0 o u t | # 1 o u t | A L U 1 | A L U 0 | M r e a d | M e m W r i t e | I m m |
|------------------|------------------|------------------|-----------------------|-----------------------|-------------|-----------------------|---------------------------------|---------------------------------|----------------------------|------------------|---------------------------------|---------------------------------|---------------------------------|-----------------------|-----------------------|------------------|------------------|-----------------------|--------------------------------------|-------------|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |

[3] What do the following acronyms stand for?

SMT

MIMD

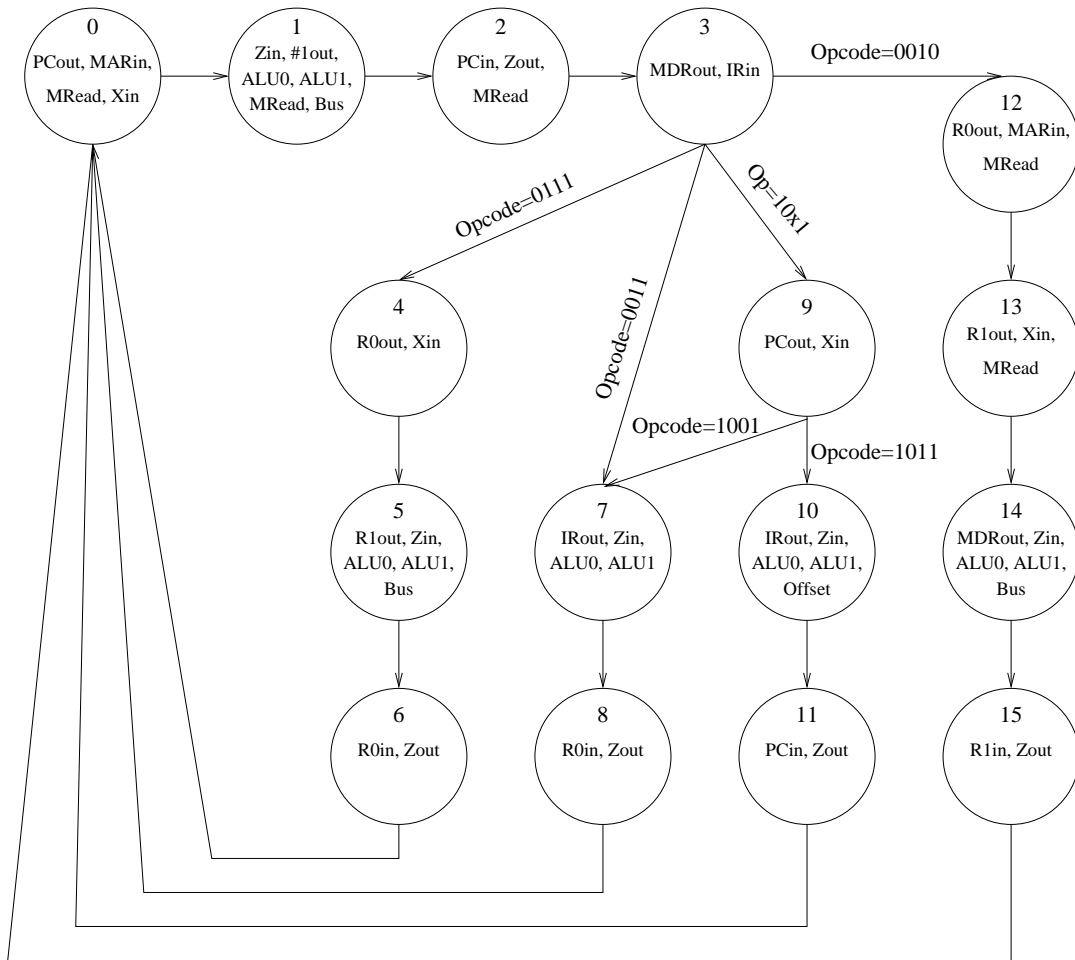
NUMA

WAW

DSM

VLIW

Here is the state diagram for a random machine:



[2] Assuming there are 4 state variables (Y_3 - Y_0), that $\text{State0} = \overline{Y_3} * \overline{Y_2} * \overline{Y_1} * \overline{Y_0}$ (0000) and $\text{State15} = Y_3 * Y_2 * Y_1 * Y_0$ (1111), write down the exact boolean equation for the **R0out** signal.

[3] Assuming the same situation as in the previous question, write down the exact boolean equation for **NextState7**.

[2] If you were to use a simple, unoptimized microcode memory to implement the control function for this machine, how big would it be? (how many entries, how wide is each entry?) (The machine has 19 control signals.)

[5] Here is a code sequence.

load R1, 0(R10)

load R2, 4(R10)

add R3, R2, R1

store R3, 8(R10)

load R4, 10(R8)

Assuming a standard 5-stage pipeline that supports hazard detection and does 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)** Schedule the code to remove as many stalls as possible. How many NOPS are left?
- d)** There are things the static scheduler does not know which makes guaranteeing correctness difficult. What if, at execution time, R10=50 and R8=48? Will your dependency graph and ability to schedule this code change?

[12] Here is a code sequence.

and R6, R7, R4

load R2, 0(R10)

add R3, R2, R6

store R3, 20(R9)

sub R9, R7, R8

add R9, R4, R7

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)** 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?

[12] Here is a code sequence.

add R1, R2, R3

sub R3, R1, R2

Label: lw R4, 0(R10)

add R4, R5, R4

sw R4, 20(R2)

and R6, R7, R5

breq R5,R6,Label ; branch to Label if R5-R6=0 (R5=R6)

Assuming a 6-stage pipeline (F,D,E,M1,M2,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 on a read returns at the end of M2.

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 memory. How many NOPS are left?

[6] Suppose I have a 4-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 3, then 0, then 1

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

The number of independent instructions Thread C can find (in order): 1, then 1, then 1, then 1

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

| Time | Slot1 | Slot2 | Slot3 | Slot4 |
|-------------|--------------|--------------|--------------|--------------|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

Now fill in the following table assuming the use of course-grained scheduling.

| Time | Slot1 | Slot2 | Slot3 | Slot4 |
|-------------|--------------|--------------|--------------|--------------|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

Now, repeat the process assuming simultaneous multithreading is being used.

| Time | Slot1 | Slot2 | Slot3 | Slot4 |
|-------------|--------------|--------------|--------------|--------------|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

[5] 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+1 experiences a fault in the M1 stage (page fault, for example):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | | | |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 | | | | |
|-----|----|----|----|----|----|----|----|----------------------------|--|----|----------------------------|----|----|----|
| i | IF | ID | RR | EX | M1 | M2 | WB | <- M1 stage has page fault | | | | | | |
| i+1 | | IF | ID | RR | EX | M1 | M2 | WB | <- Inst Decode has Illegal Instruction | | | | | |
| i+2 | | | IF | ID | RR | EX | M1 | M2 | WB | | | | | |
| i+3 | | | | IF | ID | RR | EX | M1 | M2 | WB | <- IF stage has page fault | | | |
| 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? Explain how you got your answer.