Q1. Observe the following MIPS code.

a.

```
FACT:   sw   $ra, 4($sp)
        sw   $a0, 0($sp)
        addi $sp, $sp, -8
        slti $t0, $a0, 1
        beq  $t0, $0, L1
        addi $v0, $0, 1
        addi $sp, $sp, 8
        jr   $ra

L1:     addi $a0, $a0, -1
        jal  FACT
        addi $sp, $sp, 8
        lw   $a0, 0($sp)
        lw   $ra, 4($sp)
        mul  $v0, $a0, $v0
        jr   $ra
```

b.

```
FACT:   addi $sp, $sp, 8
        sw   $ra, 4($sp)
        sw   $a0, 0($sp)
        add  $s0, $0, $a0
        slti $t0, $a0, 2
        beq  $t0, $0, L1
        mul  $v0, $s0, $v0
        addi $sp, $sp, -8
        jr   $ra

L1:     addi $a0, $a0, -1
        jal  FACT
        addi $v0, $0, 1
        lw   $a0, 0($sp)
        lw   $ra, 4($sp)
        addi $sp, $sp, -8
        jr   $ra
```

A. The MIPS assembly program above computes the factorial of a given input. The integer input is passed through register `$a0`, and the result is returned in register `$v0`. In the assembly code, there a few errors. Correct the MIPS errors.

B. For the recursive factorial MIPS program above, assume that the input is 4. Rewrite the factorial program to operate in a non-recursive manner. Restrict your register usage to registers `$s0`-`$s7`. What is the total number of instructions used to execute your solution from the non-recursive version versus the recursive version of the factorial program?

C. Show the contents of the stack after each function call, assuming that the input is 4.

Q2. Observe the following address.

a.	0x00020000
b.	0xFFFFFF00

A. If the PC is at address 0x00000000, how many jump instructions (no jump register instructions or branch instruction) are required to get to the target address in the table above?

B. In order to reduce the size of MIPS programs, MIPS designers have decided to cut the immediate field of I-type instructions from 16 bits to 8 bits. If the PC is at address 0x00000000, how many branch instructions are needed to set the PC to the address in the table above?

Q3. For the following.

	A	B
a.	33	55
b.	8a	6d

A. One possible performance enhancement is to do a shift and add instead of an actual multiplication. Since 9 × 6, for example, can be written as (2 × 2 × 2 + 1) × 6, we can calculate 9 × 6 by shifting 6 to the left 3 times and then adding 6 to that result. Show the best way to calculate A × B using shifts and adds/subtracts. Assume that A and B are 8-bit unsigned integers.

B. Write an MIPS assembly language program that performs a multiplication on signed integers using shits and adds, using the approach described above.

C. Show the best way to calculate A × B using shits and add, if A and B are 8-bit signed integers stored in sign-magnitude format.

Q4. For the following:

a.	1/3
b.	1/10

A. Write down the bit pattern in the mantissa assuming a floating point format that uses binary numbers in the mantissa. Assume there are 24 bits, and you do not need to normalize. Is this representation exact?

B. Write down the bit pattern in the mantissa assuming a floating point format that uses Binary Coded Decimal (base 10) numbers in the mantissa instead of base 2. Assume there are 24 bits, and you do not need to normalize. Is this representation exact?

Q5. For the following codes:

	Instruction sequence

a.	`SW  R16, 12(R6)`
	`LW  R16, 8(R6)`
	`BEQ R5, R4, Label  ; Assume R5!= R4`
	`ADD R5, R1, R4`
	`SLT R5, R15, R4`

b.	`SW  R2, 0(R3)`
	`OR  R1, R2, R3`
	`BEQ R2, R0, Label  ; Assume R2 == R0`
	`OR  R2, R2, R0`
	`Label: ADD R1, R4, R3`

A. Assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we only have one memory (for both instructions and data), there is a structural hazard every time we need ti fetch an instruction in the same cycles in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data. What is the total execution time of this instruction sequence in the 5-stage pipeline that only has one memory? We have seen that data hazards can be eliminated by adding NOPs to the code. Can you do the same with this structural hazard? Why?

B. Assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. If we change load/store instructions to use a register (without an offset) as the address, these instructions no longer need to use the ALU. As a result, MEM and EX stages can be overlapped and the pipeline has only 4 stages. Change this code to accommodate this changes ISA. Assuming this change does not affect clock cycle time, what speedup is achieved in this instruction sequence?

C. Assuming stall-on-branch and no delay slots, what speedup is achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

Q6. For the following:

	Pipeline Depth	Issue Width
a.	15	2
b.	30	8

A. How many register read ports should the processor have to avoid, so that there is no resource hazard due to register reads?

B. If there are no branch mispredictions and no data dependences, what is the expected performance improvement over a 1-issue processor with the classical 5-stage pipeline? Assume that the clock cycle time decreases in proportion to the number of pipeline stages.

Q7. To support multiple virtual machines, two levels of memory virtualization are needed. Each virtual machine still controls the mapping of virtual address (VA) to physical address (PA), while the hypervisor maps the PA of each virtual machine to the actual machine address (MA). To accelerate such mappings, a software approach called "shadow paging" duplicates each virtual machine's page tables in the hypervisor, and intercepts VA to PA mapping changes to keep both copies consistent. To remove the complexity of shadow pages tables, a hardware approach called nested page table (or extended page table) explicitly supports two classes of page table (VA→PA and PA→MA) and can walk such tables purely in hardware. Consider the following sequence of operations:

```
(1) Create process; (2) TLB miss; Page Fault; (4) Context switch
```

A. What would happen for the given operation sequence for shadow page table and nested page table, respectively?

B. Assuming an x86-based 4-level page table in both guest and nested page table, how many memory references are needed to service a TLB miss for native versus nested page table?

C. Among TLB miss rate, TLB miss latency, page fault rate, and page fault handler latency, which metrics are more important for shadow page table? Which are important for nested page table?

Q8. For the following:

a.	3, 180, 43, 2, 191, 88, 190, 14, 181, 44, 186, 253
b.	21, 166, 201, 143, 61, 166, 62, 133, 111, 143, 144, 61

A. For each of these 32-bit memory references listed above, identify the binary address, the tag, and the index given a direct-mapped cache with 16 one-word blocks. Also list if each reference is a hit or a miss, assuming the cache is initially empty.

B. For each of these 32-bit memory references listed above, identify the binary address, the tag, and the index given a direct-mapped cache with two-word blocks and a total size of 8 blocks. Also list is each reference is a hit or a miss, assuming the cache is initially empty.

Q9. Direct Memory Access (DMA) allows devices to access memory directly rather than working through the CPU. This can dramatically speed up the performance of peripherals, but adds complexity to memory system implementations. Explore DMA implications by answering the questions about the following peripherals.

a.	Mouse Controller
b.	Ethernet Controller

A. Does the CPU relinquish control of memory when DMA is active? For example, can a peripheral simply communicate with memory directly, avoiding the CPU completely?

B. Of the peripherals listed in the above table, which would benefit from DMA? What criteria determine if DMA is appropriate?

Q3. Of the peripherals listed in the above table, which could cause coherency problems with cache contents? What criteria determine I coherency issues must be addressed?

Q10. Benchmarks play an important role in evaluating and selecting peripheral devices, For benchmarks to be useful, they must exhibit properties similar to those experienced by a device under normal use. Explore benchmarks and device selection by answering questions about the following applications.

a.	Mathematical Computations
b.	Online Chat

A. For each application in the table, define characteristics that a set of benchmarks should exhibit when evaluating an I/O subsystem.

B. Does it make sense to evaluate an I/O subsystem outside the larger system it is a part of? How about evaluating a CPU?