Instructions

For the Breadth Exam, answer questions 1 through 4. For the Depth Exam, answer questions 1 through 7. The questions are quite specific. If, however, some confusion should arise, be sure to state all your assumptions explicitly.

Breadth Exam

(1) Caches and Virtual Memory

a) What is a *virtual address synonym* or *alias*? How can they occur?

b) Why can synonyms cause problems in a cache indexed using virtual addresses?

c) Consider a 64-kilobyte, 4-way set-associative cache, with 128-byte blocks. The cache tags are derived from the physical address. We wish to access the 64-entry 2-way set-associative translation lookaside buffer (TLB) *in parallel* with this cache. How large must the page size be to prevent synonyms from being a problem? SHOW YOUR WORK.

d) Sketch the cache and TLB. Be sure to show the number of bits in each field in the cache and TLB. Show the comparators. Be sure to explicitly label which bits from the addresses are used on each signal (i.e., which bits are used to index the cache and TLB, which bits are used for the tags, etc.). Assume the virtual and physical addresses each contain 32 bits, and are labeled VA<31:0> and PA<31:0> respectively.

(2)Floating-point numbers

A floating-point format specifies how bits from a word are interpreted. Often bits are partitioned into sign, mantissa, and exponent fields and the radix is not explicitly represented.

a) In some floating-pointing formats, such as IEEE single- and double-precision, the most significant bit of the mantissa is not explicitly represented. For what choices of format is this optimization not possible? What number(s) require special interpretation with this optimization? Explain.

b) Some floating-pointing formats, such as IBM 360/370, use radix sixteen rather than radix two. Give one important implementation advantage of radix sixteen over radix two.

c) Most floating-point adders can shift one input mantissa (and adjust its exponent) before adding and can shift the resulting mantissa after adding. Why might both these shifts be necessary?

(3) Instruction Pipelines

The typical instruction pipeline in most modern microprocessors with a load/store instruction set is as follows:

where the stages are:

IF:	Instruction Fetch
ID:	Instruction Decode
EX:	Execute in ALU
MA:	Memory Access
WB:	Write Back

a) List the two most important factors that degrade the throughput for such instruction pipelines. For each of the factors, suggest some ways of overcoming the degradation.

b) What advances must be made in hardware and/or software to efficiently utilize an instruction pipeline of high degree, e.g., 8 segments or higher?

(4) Processor and Memory Performance

a) The base CPI (clocks per instruction) for a 32-bit CPU with a perfect memory system (all memory references are serviced in a single cycle) is 1.5. Find the CPI with a memory system containing a single 16KB direct-mapped, write-back (copy-back) cache with a miss ratio of 0.05.

Make the following assumptions. Since instructions and data references share the same cache, all data references stall for one cycle to gain access to the cache. 15% of the instructions are loads, and 6% are stores. There are 16 bytes per line (block). The cache fetches words of a line in address order, and the CPU stalls until all the words of the block arrive. There is no write buffer. Assume the memory latency is 6 clocks, the transfer rate is 4 bytes per clock cycle, and that 50% of the lines (blocks) in the cache are dirty. (The latency is the time from the start of the request until the first word has been *received*).

b) Two schemes for improving the performance of the memory system are suggested. Which one has a higher impact on performance? Why?

(i) The accessed word is fetched first on a cache miss, and the CPU is stalled only until the accessed word arrives. Assume that the CPU does not execute any other load or store instructions, i.e., has no need for the cache, until the remaining words have arrived in the cache.

(ii) A large write buffer is inserted so that write backs to memory do not cause any stalls.

Depth Exam

(5) An addressing mode has a *side effect* if it causes some change to the architectural state of the machine in addition to its primary function. For example, an autoincrement mode on a register causes the value of the register to be updated, regardless of the other effects of the instruction that uses the addressing mode. Therefore it has a side effect.

The VAX-11 architecture, designed in the 1970s has a very rich set of addressing modes, many of which have side effects. In addition, almost any addressing mode can be used in conjunction with any of the 16 general-purpose registers. For example, the autoincrement addressing mode can be used with R15, the program counter.

Load/Store architectures, such as the MIPS R2000 and the Sun SPARC, developed in the 1980s, have very few addressing modes. Moreover, the limited addressing modes in these machines do not have side effects. For example, the R2000 has only a displacement (register + offset) mode, and the SPARC uses both displacement and an index (register + register) mode. The more ‘‘complex’’ modes with side effects, such as autoincrement, are discarded completely.

More recently, the IBM RS/6000 and the HP Snake series of machines have reintroduced the use of some ‘‘complex’’ addressing modes with side effects. For example, the HP Snake has the autoincrement addressing mode, and the IBM RS/6000 has a load update addressing mode (which can be viewed as a generalization of the autoincrement mode), with their load/store instructions (the architectures are still register-register load/store architectures).

What are the reasons that addressing modes with side effects, such as autoincrement, were discarded in the earlier load/store architectures, and why have they reappeared in some of the more recent ones? Which other addressing modes do you think might appear? Why, or why not?

(6) Cache memory is effective in a uniprocessor because it exploits temporal and spatial locality.

a) Explain how a cache exploits temporal locality.

b) Explain how a cache exploits spatial locality.

c) What are the implications of these two types of locality for implementing a shared-bus, shared-memory multiprocessor, i.e., a "multi"? If workload is important, explain how.

(7) While several multiple-instruction-multiple-data (MIMD) parallel computers have been designed using today's powerful microprocessors, single-instruction-multiple-data (SIMD) machines tend to use custom logic for instruction fetch and control. There are at least two ways to use microprocessors in a SIMD machine:

Method 1: Build logic to respond to each microprocessor's instruction fetch with the correct instruction regardless of the instruction fetch address.

Method 2: Have every microprocessor execute an identical copy of the program.

a) Discuss the problems that must be overcome to use Method 1.

b) Discuss the problems that must be overcome to use Method 2.

c) Discuss the relative merits of building a SIMD machine with Method 1, Method 2, or the traditional method of using custom logic for instruction fetch and control.