These are questions taken from exams I have given in CSE 378 in the past. I would like you to answer them and hand in your solutions, so that both of us can gauge the level of your background in computer architecture.

Try to answer the questions as best you can. You are welcome to use either the undergraduate book (Hennessy & Patterson, *Computer Organization & Design: The Hardware/Software Interface*) or the P548 book (same authors, *Computer Architecture: A Quantitative Approach)* to remind you of the technical material. It is not so important that you get the details right, but that you have some clue as to what the issues are and how to solve the problems.

This will be a hard test for you, especially if you have never had a course in computer architecture. If the exam takes you more than two hours, even using the book, get in touch with me so we can talk about whether you should take P548 and, if you do, how you can bolster your architecture background. For example, if you don't know what pipelining is, you probably won't be able to handle question 3. But your unfamiliarity will tell us that you'll need to pay particular attention to my pipelining review in class and do some reading in the undergraduate text.

Last time I taught P548, several students had trouble with this exam. All borrowed and read sections of the undergraduate text as the course progressed, and all did just fine in the class. I expect the same will be true for you.

We will grade your solutions, but they will not count toward your grade in P548.

Good luck!

1) [8 points] [9 points] "***Design for the common case***" is one of the architectural design principles. It means that architects should support in an architecture only the behavior that occurs most commonly in executing programs. Name ***three*** different ways in which the MIPS architects followed this principle. For each one, first say (1) what the common case is and then (2) what part of the MIPS ***architecture*** supports it.

   Be sure to discuss *different* aspects of program behavior. Repeating the same common program behavior and using different architectural designs will not be allowed. For example, one type of program behavior is that programs execute a subset of operations most frequently. This can lead to a couple of different architectural designs: few, simple instructions; few formats; and few format fields. To get credit, you could only use one of these three alternatives. Of course, now that I've mentioned this one, it's off the table.

   If you are not familiar with the MIPS architecture, pick one of your own. (Hint: a RISC architecture will be easier to discuss than a CISC architecture.)

2a) [2 points] MIPS uses a signed 16-bit value as the branch offset for the PC-relative addressing mode. If the offset represents byte addresses, what is the branching address range?

2b) [2 points] How could you access a memory location outside this range on a conditional branch?

3) The following table names and describes the pipeline stages in the Alpha 21164 pipeline.

Stage 0	Instruction fetch Dynamic branch prediction
Stage 1	If predict taken, redirect the fetch Target address calculation Decode Instruction TLB check
Stage 2	Data dependence checking for instructions within the same superscalar issue slot
Stage 3	Data dependence checking for instructions in different superscalar issue slots Register read Instruction issue
Stage 4	Arithmetic execution Effective address calculation
Stage 5	Branch resolution Data cache access
Stage 6	Register write

3a) [2 points] What is the penalty (if any) for fetching a correctly predicted target instruction in this pipeline?

3b) [2 points] What is the penalty (if any) for fetching a correctly predicted sequential instruction in this pipeline?

3c) [2 points] What is the penalty (if any) for mispredicting the next instruction to fetch in this pipeline?

3d) [3 points] What is the load-use penalty (if any) in this pipeline, assuming as much bypass logic as you wish? At the end of which stage is the load data available? At the beginning of which stage is it needed?

4a) [2 points] What is latency?

4b) [2 points] What is throughput?

4c) [9 points] Three different techniques that affect performance are listed below. For each one, say whether they increase, decrease or do not affect the latency, or increase, decrease or do not affect throughput. Those techniques that both decrease latency and increase throughput (which are, of course, both advantages) must have some other disadvantage not related to performance. What is it?

2) Technique	3) Latency	4) Throughput	5) A disadvantage?
6) superscalars	7)	8)	9)
10) smaller pages	11)	12)	13)
14) wider processor-memory bus	15)	16)	17)

5a) [2 points] Caching and paging have a common underlying principle that enables them to provide better performance. What is it?

5b) [9 points] What are three differences between caches and virtual memory? Do not list any differences that are related to the size of any cache or memory configuration. Examples of disallowed differences are caches are smaller than memory and blocks are smaller than pages.

6) [4 points] A page table for a virtual address space of 2^{32} bytes requires a fairly large amount of memory. One technique for reducing the size of the page table is to have a hierarchy of page tables, in which the entries in the first level in the hierarchy (called the *level-1 page table*) point to individual page tables in the second level of the hierarchy (called the *level-2 page tables*). Entries in the second-level page tables contain a page frame number for the page (as well as the other information normally found in a page table entry). To conserve the amount of page table space that must always reside in physical memory, only the level-1 page table always resides in physical memory, and the level-2 page tables can be paged.

i) To implement the hierarchy of page tables, the virtual page number can be divided into two parts, a level-1 page table offset and a level-2 page table offset, as shown in the virtual address below.

level-1 page table offset	level-2 page table offset	page offset

The level-1 page table offset indexes into the level-1 page table (a processor register points to the beginning of the level-1 page table). The indexed entry contains the physical address for the beginning of one of the level-2 page tables, assuming this level-2 page table resides in memory (if not, a page fault will occur). The level-2 page table offset is added to this physical address to index into the level-2 page table. This entry provides the physical address (page frame number) for the page that contains the instruction or data the CPU has accessed.

Assume that:

- the level-1 page table occupies exactly one page of memory,
- the virtual address is 32 bits,
- page size is 4 KB,
- the PTE is 4 bytes.

6a) [3 points] How many bits are in each of the three fields in the virtual address?

6b) [2 points] How many entries are in the level-1 page table?

6c) [2 points] How many entries are in each level-2 page table?

6d) [2 points] How many pages do the level-2 page tables map?

7) [12 points] Execution time can be measured as: instructions/program * average cycles/ instruction * time/cycle. For each of the following design changes and each factor in the equation, say whether, all other things being equal, the change cannot affect the factor, will tend to increase it, or decrease it. Give a justification or explanation for your choice if you answer increase or decrease. To get full credit for any part, you need to get ***both*** the type of change, if any, and the reason correct.

- A superpipelined implementation (more pipelined stages)
  - instruc/program
  - cycles/instruc
  - time/cycle

- A larger cache
  - instruc/program
  - cycles/instruc
  - time/cycle