1. [6 points]
   (i) (4 points) Control hazards are a major hurdle to pipeline-based performance enhancement. One technique used to deal with control hazards is the delayed branch. With the help of an example code and an example pipeline timing diagram, explain how delayed branch helps in reducing the performance penalty due to control hazards.

   The delayed branch technique involves changing the semantics of a branch instruction such that one (or more) instruction following the branch is not conditional on the outcome of the branch. That is, instructions in the delay slot(s) must be executed irrespective of the branch outcome. The compiler is responsible for filling the delay slot(s) with useful instructions; if no useful instruction can be found, it places a "nop" instruction.

   Consider the example code given below.

   With regular branch                   With delayed branch
   add         r3, r2, r1                  slt         r5, r4, r6
   slt         r5, r4, r6                  bne         label
   bne         label                      add         r3, r2, r1
   sub         r7, r8, r9                 sub         r7, r8, r9
   .                         .                         
   .                         .                         
   label:         and         r7, r8, r9         label:         and         r7, r8, r9

   Now, consider the standard MIPS 5-stage pipeline: IF, ID, EX, MEM, WB. Assume that branch outcomes are known in the ID stage. In the first case (the one with the regular branch), there is a control hazard while fetching the sub instruction because the outcome of the bne instruction is not known. In the second case, there is no control hazard while fetching the sub instruction, because the outcome of the bne instruction is known (assuming the branch outcome is determined in the ID stage itself). Notice also that the add instruction does not encounter a control hazard, as it is no longer control-dependent on the bne instruction.

   With regular branch
   add         IF         ID         EX         MEM         WB
   slt         IF         ID         EX         MEM         WB
   bne         IF         ID         EX         MEM         WB
   sub         IF         -         -         -         -
                 |                 |                 |
   Does not know if sub is needed
With delayed branch

<table>
<thead>
<tr>
<th></th>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>slt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bne</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub/and</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Knows if sub or and is needed

(ii) (2 points) Describe two drawbacks of the *delayed branch* technique.

1. *The delayed branch feature becomes part of the instruction set architecture, and so creates object code compatibility issues.*

2. *Many times, the compiler may not find a useful instruction to place in the delay slot, and so is forced to place a "nop" instruction in the delay slot.*

3. *Implementing the delayed branch requires additional hardware in the processor.*

4. *Modern pipelines are long, and having a 1 cycle delay slot may not make a significant improvement.*
2. [7 points]
(i) (3 points) Explain how multitasking helps improve the performance of a computer system. Give one reason why multitasking might reduce the performance of a computer system.

Multitasking can help improve the performance of a computer system by increasing the utilization of the processor. When a running process is blocked due to an I/O event, for instance, multitasking allows another process to run on the processor, thereby increasing the throughput.

Multitasking can hurt performance if the time required to do a context switch is too large. Also, if too many processes are active in a system, the memory system performance deteriorates because of poor locality.

(ii) (4 points) List two similarities and two differences between cache memory and paging-based virtual memory.

**Similarities:**

1. Both of them serve as fast storage for the next lower level of the memory hierarchy—the cache memory for the main memory and virtual memory system for the hard disk.

2. Both of them use fixed-size blocks/pages for managing storage and for transferring data from the lower hierarchy level.

**Differences:**

1. Cache misses are handled by hardware (cache controller), whereas page misses are handled by software (operating system).

2. Cache memory uses direct mapping or a low level of associative mapping, whereas virtual memory uses a high level of associative mapping such as fully-associative mapping.
3. [7 points]
   (i) (4 points) List two similarities and two differences between paging and segmentation.

   **Similarities:**

   1. Both paging and segmentation are techniques to implement virtual memory; both involve partitioning the memory address space into blocks, and allow major portions of the address space to be not present in physical memory at any time. When the processor encounters a virtual address, a translation is done to determine the corresponding physical address.

   2. Both schemes can be used to implement protection.

   **Differences:**

   1. The paging scheme has only one linear address space, whereas the segmentation scheme has multiple address spaces.

   2. Paging is not a part of the instruction set architecture, and so is transparent to the assembly language programmer (or compiler). Segmentation, on the other hand, is a part of the instruction set architecture, and so is not transparent to the assembly language programmer (or compiler).

   3. The pages in a paging scheme have fixed size, whereas the segments in a segmentation scheme can have variable sizes.

   (ii) (3 points)
   Describe how a **hierarchical page table** is structured and accessed. Explain one advantage and one disadvantage of using an hierarchical page table.

   A hierarchical page table uses multiple levels of (small) page tables that are organized as a hierarchy. The page table at the top level of the hierarchy holds pointers to the next-level page tables, which themselves hold pointers to the next level, and so on. The page tables at the lowest level of the hierarchy store the physical frame numbers of the virtual pages.

   The advantage of splitting a large page table into a hierarchical one is that memory needs to be allocated only for portions of the page table that are currently in use. (Even if a computer system supports a huge virtual address space, a process may be using only a small portion of that address space.)

   The disadvantage of using an hierarchical page table is that when there is a TLB miss, the virtual-to-physical address translation would require accessing multiple page tables, increasing the translation time.