Instructions:

- Print your name in the space provided below.
- This examination is closed book and closed notes, aside from the permitted one-page formula sheet and the MIPS reference card. No calculators or other computing devices may be used.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 8 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name ____________________________

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_______________________________
signed
1. Consider executing the following sequence of instructions in the pipelined MIPS datapath:

```
add   $s0, $s1, $s2   # stmt 1
lw    $s5, 0($s0)     # stmt 2
add   $s3, $s4, $s7   # stmt 3
sw    $s3, 0($s0)     # stmt 4
```

a) [10 points] Which, if any, of the instructions will require forwarding of an operand (but not a stall)? Identify each such instruction (by number), and the specific register that must be forwarded.

- **$s0 must be forwarded from stmt 1 to stmt 2.**
- **$s3 must be forwarded from stmt 3 to stmt 4.**

b) [10 points] Which, if any, of the instructions will require a stall (and possibly a forwarding operation)? Identify each such instruction (by number), and explain why forwarding alone is not sufficient.

If the `add` command in stmt 3 used the destination register for the preceding `lw` command, we would need to stall the `add`, but it doesn't.

c) [6 points] How many clock cycles will be required in order to complete the execution of the given sequence of instructions?

The first instruction will complete after 5 cycles, and one more instruction will complete on each subsequent clock cycle until all have been completed, for a total of 8 clock cycles.

If stalls had been needed, then each stall would have added 1 clock cycle to the total.
For question 2, refer to the copy of Fig 4.51 from P&H that was distributed along with the test.

2. a) [10 points] Locate the RegDst control signal labeled A on the diagram. The signal is passed through the ID/EX inter-stage buffer. What undesirable thing(s) could occur if the RegDst control signal were passed directly from the Control unit to the MUX, bypassing the inter-stage buffer? Or would that be perfectly OK?

Suppose that instruction I1 is entering the ID stage and instruction I0 is entering the EX stage.

When I1 is decoded, an appropriate value will be set for RegDst.

If that value is not buffered in the ID/EX inter-stage buffer, then it would reach the MUX in the EX stage while I0 is still in that stage, which could cause the wrong write register number to be passed forward to subsequent stages.

If I0 is an R-type or lw instruction, and I1 is not the same type of instruction, this could cause the wrong register to be written to when I0 reaches the WB stage of the pipeline.

b) [10 points] Consider the MEM/WB inter-stage buffer, labeled B on the diagram. How many bits of storage must this inter-stage buffer provide? Write a list of the items that must be stored there, and give the total number of bits needed to do so.

There are five output lines from the MEM/WB inter-stage buffer. Taking them from the top, they are:

<table>
<thead>
<tr>
<th>Bits</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RegWrite</td>
</tr>
<tr>
<td>1</td>
<td>MemtoReg</td>
</tr>
<tr>
<td>32</td>
<td>value from Read data port of the Data Memory unit</td>
</tr>
<tr>
<td>32</td>
<td>value computed by the ALU</td>
</tr>
<tr>
<td>5</td>
<td>write register number</td>
</tr>
</tbody>
</table>

So, that requires a total of 71 bits of storage.
3. [10 points] There are three types of cache misses: conflict misses, cold misses and compulsory misses. Describe two of the types in detail (your choice).

**conflict miss:**
- a miss in a direct-mapped or set-associative cache due to the "collision" of two different blocks in the same cache location, so that the needed block has been replaced by another block mapped to the same cache location, and hence the needed block must be fetched from memory again; these do not occur in a fully-associative cache of the same capacity.

**cold or compulsory miss:**
- a miss caused by the first access to a block (that has never been in the cache)

**capacity miss:**
- a miss caused by the inability of a cache to store all of the blocks needed during the execution of a program; occur when blocks are replaced and then later retrieved (again)

4. [10 points] Is the effectiveness of a cache dependent or independent of the characteristics of the programs that are being executed? Explain carefully.

The effectiveness of a cache depends on the extent to which it contains memory values needed by the running program.

That is determined by the extent to which the program exhibits temporal locality, and the extent to which the strategy used to populate the cache anticipates future memory accesses.

5. [8 points] You must implement a 64KB cache system for a new processor. You may choose either a direct-mapped cache, in which each line stores a single word of data, or a 4-way set-associative cache design in which each line stores four words of data. Discuss how and why the 4-way set-associative design might provide better performance. Be complete and precise.

The 4-way set-associative cache may reduce the number of conflict misses since up to 4 addresses that map into the same set may be stored without requiring a replacement.

All the tags for a given set are examined in parallel, so that does not add to the cost of doing a lookup. With a set size of 4, even LRU replacement is cheap enough to not hurt performance (more than the gain it achieves).

On the other hand, in a direct-mapped cache, there is only one possible location for each memory address and therefore no way to avoid direct conflict replacements.
6. [8 points] Consider two scenarios:
   
i) the running program references a physical address that is not in the cache
   ii) the running program references a virtual address that lies on a page that is not currently in physical memory

   Compare the relative effect of the two scenarios on the running program. (Do not describe the detailed mechanics of how a cache miss or a page fault are handled.)

   **In scenario i, we have a simple cache miss, requiring fetching data from memory, which would typically cost a few hundred clock cycles at worst.**

   **In scenario ii, we must fetch the referenced page from the swap space on disk, and that will typically require millions of machine cycles.**

   **So, accessing main memory on a cache miss is MUCH faster than accessing the hard disk to fetch a virtual page.**

   **So, the second scenario will result in a much greater time penalty for the running program.**

7. [8 points] Describe the role of the translation lookaside buffer (TLB) in a modern computer system.

   **The TLB is used to cache recently-accessed page table entries, which are used when translating virtual addresses to physical addresses.**

   **The TLB is vital in order for this address translation to be fast enough to make virtual memory systems practical.**
8. [10 points] Why is direct memory access (DMA) an improvement over CPU programmed I/O?

CPU programmed I/O requires the direct involvement of the CPU to execute instructions that enable transfers between I/O devices and primary memory (DRAM). This may involve polling, or more likely, frequent interrupts to the processor.

DMA allows such transfers to occur with CPU involvement only to initiate and deal with the completion of such data transfers.

So, DMA offloads much of the cost of a transfer of data between memory and an I/O device from the processor to the DMA controller.

This may make the data transfer very slightly faster.

The main benefit is that this frees the CPU for use in executing code for other programs (or the OS itself).