Problem 1: The pipeline execution diagram below shows two floating-point instructions on the
Chapter-3 implementation of DLX, where the divide unit, which is not pipelined, has latency 24
and initiation interval 24, and the multiply unit, which is fully pipelined, has a latency of 6 and an
initiation interval of 1.

\[
\begin{align*}
div & \quad \text{IF ID D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 ... D24 MEM WB} \\
\text{mul} & \quad \text{IF ID M1 M2 M3 M4 M5 M6 M7 MEM WB}
\end{align*}
\]

Finding structural hazards with the notation above is tricky because there is no indication that D1
and D2 refer to the same piece of hardware while M1 and M2 refer to different hardware. (The
code above does not encounter a structural hazard; it would if the second instruction were also a
divide.) Develop a notation that fixes the problem and can be used for any latency and initiation
interval. Use the notation in the pipeline execution diagram for problem 2.

Problem 2: An implementation of DLX performs in-order execution (but out-of-order comple-
tion), is fully bypassed, and properly interlocked (the implementation discussed in Chapter 3).
MEM-stage structural hazards are resolved by stalling just before the MEM stage. WAW haz-
rards are handled by nulling the earlier instruction. The floating-point functional units perform the
operations as described in Chapter 3, however the timings are as described below:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Init. Inter.</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIV</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>MUL</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>ADD</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Find the pipeline execution diagram for the following code on this system

\[
\begin{align*}
div & \quad f3, f4, f5 \\
\text{mul} & \quad f0, f1, f2 \\
\text{mul} & \quad f3, f6, f7 \\
\text{sub} & \quad f8, f9, f10 \\
\text{mul} & \quad f11, f0, f12
\end{align*}
\]

Problem 3: Repeat the problem above on an implementation that is the same as the one above
except MEM-stage structural hazards are resolved by stalling in ID.

Problem 4: The program below runs on a DLX implementation that uses dynamic scheduling
with Tomasulo’s algorithm and register renaming. The multiply unit has a latency of 8 and an
initiation interval of 1, and has two reservation stations, numbered 1 and 2. Branch targets are
computed in the ID stage.

Show the execution of the code below up to the second writeback of the multiply instruc-
tion assuming (as we have so far) there are no cache misses.

\[
\begin{align*}
\text{addi} & \quad r1, r0, \#1000 \\
\text{LOOP:} & \\
\text{ld} & \quad f1, 1024(r1) \\
\text{subi} & \quad r1, r1, \#8 \\
\text{mul} & \quad f0, f0, f1 \\
\text{bneq} & \quad r1, \text{LOOP}
\end{align*}
\]

Problem 5: For the problem above, how large can multiply’s latency be (with an initiation interval
of 1) without running out of reservation stations after some number of iterations?